SIEMENS

SIMATIC

SCL per S7-300/400
Programmazione di blocchi

Manuale

Numero di ordinazione del manuale:
6ES7811-1CA02-8EA0

Avvertenze tecniche di sicurezza

Il presente manuale contiene avvertenze tecniche relative alla sicurezza delle persone e alla prevenzione dei danni materiali che vanno assolutamente osservate. Le avvertenze sono contrassegnate da un triangolo e, a seconda del grado di pericolo, rappresentate nel modo seguente:

Pericolo di morte

significa che la non osservanza delle relative misure di sicurezza **provoca** la morte, gravi lesioni alle persone e ingenti danni materiali.

Pericolo

significa che la non osservanza delle relative misure di sicurezza **può causare** la morte, gravi lesioni alle persone e ingenti danni materiali.

Attenzione

significa che la non osservanza delle relative misure di sicurezza **può causare** leggere lesioni alle persone o lievi danni materiali.

Avvertenza

è una informazione importante sul prodotto, sull'uso dello stesso o su quelle parti della documentazione su cui si deve prestare una particolare attenzione.

Personale qualificato

La messa in servizio ed il funzionamento del dispositivo devono essere effettuati solo in base al manuale.

Interventi nel dispositivo vanno effettuati esclusivamente da **personale qualificato**. Per personale qualificato ai sensi delle avvertenze di sicurezza contenute nella presente documentazione si intende chi dispone della qualifica a inserire, mettere a terra e contrassegnare, secondo gli standard della tecnica di sicurezza, apparecchi, sistemi e circuiti elettrici.

Uso conforme alle disposizioni

Osservare quanto segue:

Pericolo

Il dispositivo deve essere impiegato solo per l'uso previsto nel catalogo e nella descrizione tecnica e solo in connessione con apparecchiature e componenti esterni omologati dalla Siemens.

Marchi di prodotto

SIMATIC®, SIMATC HMI® e SIMATIC NET® sono marchi registrati della SIEMENS AG.

Tutte le altre sigle qui riportate possono corrispondere a marchi il cui uso, da parte di terzi, può violare i diritti di proprietà.

Copyright © Siemens AG 1998 All rights reserved

La duplicazione e la cessione della presente documentazione sono vietate, come pure l'uso improprio del suo contenuto, se non dietro autorizzazione scritta. Le trasgressioni sono passibili di risarcimento dei danni. Tutti i diritti sono riservati, in particolare quelli relativi ai brevetti e ai marchi registrati.

Siemens AG
Bereich Automatisierungs- und Antriebstechnik
Geschaetfsgebiet Industrie-Automatisierungssysteme
Postfach 4848, D-90327 Nuernberg

Esclusione della responsabilità

Abbiamo controllato che il contenuto della presente documentazione corrisponda all'hardware e al software descritti. Non potendo tuttavia escludere eventuali divergenze, non garantiamo una concordanza totale. Il contenuto della presente documentazione viene comunque verificato regolarmente e le correzioni o modifiche eventualmente necessarie sono contenute nelle edizioni successive. Saremmo lieti di ricevere qualsiasi proposta di miglioramento.

© Siemens AG 1998
Ci riserviamo eventuali modifiche

Siemens Aktiengesellschaft

6ES7811-1CA02-8EA0

Prefazione

Scopo del manuale

Lo scopo del presente manuale è quello di aiutare l'utente nella stesura di programmi utente nel linguaggio di programmazione SCL. Vengono descritte le procedure fondamentali per la stesura di programmi con l'editor SCL, il compiler SCL e il debugger SCL.

Inoltre, il manuale contiene una parte di riferimento sugli elementi linguistici del linguaggio di programmazione SCL. Vengono descritte la sintassi e le modalità di funzionamento dei singoli elementi linguistici.

Destinatari

Il presente manuale si rivolge a programmatori che scrivono programmi S7, agli addetti alla messa in servizio ed al personale di servizio. Il possesso di nozioni generali sulle tecniche di automazione facilita notevolmente il lavoro.

Validità del manuale

Il presente manuale è valido per il software di programmazione STEP 7 Versione 3.0. Esso è valido per il pacchetto STEP 7 Software di base.

Conformità alla norma IEC 1131-3

SCL è conforme al linguaggio "Structured Control Language", definito nella norma DIN EN-61131-3 (int. IEC 1131-3), pur essendoci sostanziali differenze concernenti le operazioni. Per informazioni dettagliate sulla conformità alla norma si rimanda alla tabella di normalizzazione contenuta nel file NORM.TAB di STEP 7.

Panoramica sulla documentazione

Come supporto per la configurazione e la programmazione di un sistema di automazione S7, l'utente può avvalersi di un'estesa documentazione specifica. Le spiegazioni e la figura seguenti facilitano la consultazione dei manuali.

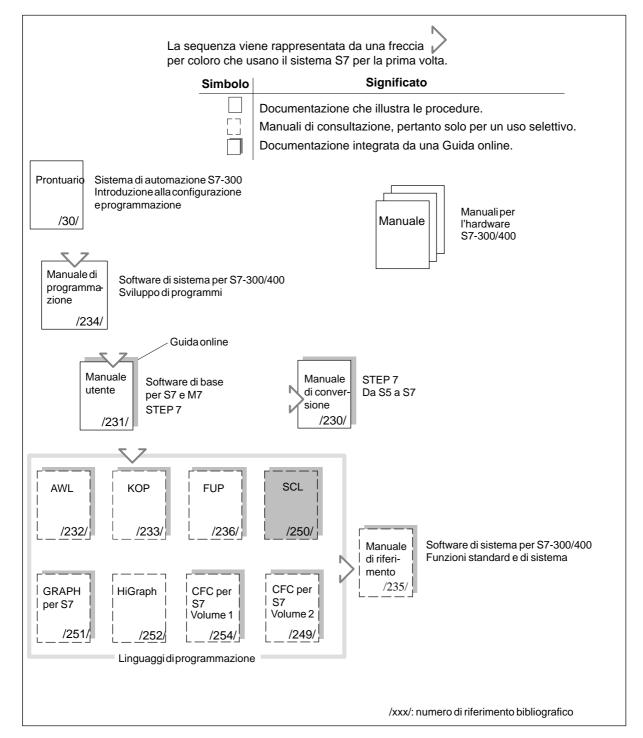



Figura 1-1 Panoramica sulla documentazione S7

Tabella 1-1 Contenuto della documentazione S7

Titolo	Contenuto
Prontuario Sistema di automazione S7-300 Introduzione alla configurazione e programmazione	Questo prontuario consente un facile approccio con le procedure di configurazione e programmazione dell'S7-300/400. E' utile soprattutto per coloro che utilizzano il sistema di automazione S7 per la prima volta.
Manuale di programmazione Software di sistema per S7-300/400 Sviluppo di programmi	Il manuale di programmazione "Software di sistema per S7-300/400 - Sviluppo di programmi" fornisce le nozioni fondamentali relative alla configurazione del sistema operativo e del programma utente di una CPU S7. Questo manuale è destinato agli utenti che non conoscono i sistemi S7-300 e S7-400. Offre una panoramica delle procedure di programmazione e consente quindi di eseguire la configurazione del programma utente.
Manuale di riferimento Software di sistema per S7-300/400 – Funzioni standard e di sistema	Il sistema operativo delle CPU dell'S-7 include funzioni di sistema e blocchi organizzativi che possono risultare utili durante la programmazione. Il manuale fornisce una panoramica delle funzioni di sistema, dei blocchi organizzativi e delle funzioni standard caricabili disponibili con l'S7, nonché, a titolo informativo, descrizioni dettagliate delle interfacce da utilizzare nel programma utente.
Manuale utente STEP 7	Il manuale utente "STEP 7" illustra l'utilizzo principale e le funzioni del software di automazione STEP 7. Il manuale fornisce, sia all'utente principiante di STEP 7 che all'esperto di STEP 5, una panoramica delle procedure di configurazione, programmazione e messa in servizio dell'S7-300/400. Durante l'esecuzione del software, è possibile consultare la Guida online.
Manuale di conversione STEP 7 Da S5 a S7	Il manuale di conversione "STEP 7. Da S5 a S7" è utile se si desidera convertire il programma S5 per poterlo eseguire sulle CPU S7. Il manuale fornisce una panoramica delle procedure del programma di conversione. Per istruzioni dettagliate sull'uso delle funzioni di conversione, consultare la Guida online, che descrive anche le interfacce delle funzioni convertite disponibili in STEP 7.
Manuali AWL, KOP, FUP, SCL[1]	I manuali dei linguaggi di programmazione AWL, KOP, FUP e SCL contengono sia le istruzioni per l'utente che la descrizione del linguaggio. Per la programmazione dell'S7-300/S7-400 è sufficiente un solo linguaggio; tuttavia, all'occorrenza, è possibile utilizzare vari linguaggi all'interno di un unico progetto. Se si usa per la prima volta un linguaggio, è consigliabile leggere il manuale per acquisire familiarità con le procedure di sviluppo dei programmi. Durante l'esecuzione del software è possibile consultare la Guida online che fornisce informazioni dettagliate sull'uso dei vari editor/compilatori.
Manuali GRAPH[1] , HiGraph[1], CFC[1]	I linguaggi GRAPH, HiGraph e CFC offrono ulteriori possibilità: permettono di creare comandi sequenziali, comandi di stato o collegamenti grafici di blocchi. I manuali contengono sia le istruzioni per l'utente che la descrizione dei linguaggi. Se si usa per la prima volta un linguaggio, è consigliabile leggere il manuale utente per acquisire familiarità con le procedure di sviluppo dei programmi. Durante l'esecuzione del software (ad eccezione di HiGraph) è possibile consultare la Guida online che fornisce informazioni dettagliate sull'uso dei vari editor/compilatori.

1 Pacchetti opzionali per il software di sistema per S7-300/400

Guida all'uso del manuale

Il presente manuale su SCL presuppone la conoscenza di nozioni teoriche sui programmi S7 descritte nel manuale di programmazione **/234/** . Poiché i pacchetti linguistici fanno riferimento al software di base STEP 7, si dovrebbe avere una buona conoscenza del software di base descritto nel manuale utente **/234/** .

Questo manuale tratta gli argomenti sotto descritti.

- Il capitolo 1 contiene informazioni generali sulla programmazione con SCL.
- Il capitolo 2 descrive, sulla base di un esempio pratico, la progettazione di un programma che potrà poi essere eseguito.
- I capitoli 3-6 descrivono l'ambiente di progettazione di SCL. L'utente apprende l'uso di Editor, Compiler e Debugger di SCL.
- I capitoli 7-19 contengono testi di consultazione, che intendono fornire dettagliate informazioni sulle singole istruzioni di SCL.

In appendice vengono descritti:

- L'intera rappresentazione della sintassi di SCL.
- Un glossario in cui vengono spiegati importanti termini specifici.
- Un indice analitico che aiuta l'utente a reperire rapidamente i passaggi del testo concernenti determinati argomenti specifici.

Convenzioni

I riferimenti ad altra documentazione vengono indicati da un numero racchiuso tra due barre /.../. In base a questo numero, nella bibliografia nell'appendice D, è possibile risalire al titolo del manuale.

Supporto

Per ulteriori informazioni sull'uso del software, che non sono contenute nella documentazione su carta o nella guida online, rivolgersi al personale di assistenza presso le filiali e le rappresentanze Siemens. Gli indirizzi si trovano nell'appendice di **/70/** e di **/100/**, nei cataloghi e nel Compuserve (`go autforum`).

Inoltre, è sempre a disposizione del cliente la nostra hotline:
Tel. + 49(911) 895-7000 (Fax 7001)

Nel caso di domande o commenti sul manuale, compilare il questionario che si trova in fondo e spedirlo all'indirizzo indicato. È gradita anche una valutazione personale del manuale stesso.

Per facilitare l'apprendimento del sistema di automazione SIMATIC S7, la Siemens organizza dei corsi specifici. Per maggiori informazioni, rivolgersi al centro di addestramento regionale o al centro di addestramento centrale in Germania:

D-90327 Norimberga, Tel. 0911 / 895 3154.

Informazioni particolari

La sezione utente di questo manuale non contiene istruzioni operative precise descritte in singole sequenze ma mira a spiegare le modalità di procedere essenziali. Informazioni più dettagliate relative alle singole finestre di dialogo del software e alla loro elaborazione si trovano nella Guida online.

Contenuto

Parte 1: Sviluppo di programmi

Presentazione del prodotto 1

Linguaggio di programmazione SCL

Sempre più spesso, i sistemi di automazione devono svolgere, oltre ai classici compiti di comando, anche compiti di gestione dati e complesse funzioni matematiche. SCL per S7-300/400 (Structured Control Language) è un linguaggio di programmazione concepito appositamente per svolgere i compiti suddetti, che facilita notevolmente la programmazione ed è normalizzato secondo IEC 113-3.

SCL offre un valido aiuto all'utente non solo per svolgere i "normali" compiti di comando, ma anche per realizzare numerose applicazioni ed è quindi molto più efficace dei "classici" linguaggi di programmazione nei seguenti campi di applicazione:

- Gestione dati
- Ottimizzazione di processo
- Gestione di ricette
- Compiti matematici/statistici.

Dati tecnici

Per poter lavorare con SCL è necessario disporre di un dispositivo di programmazione SIMATIC o di un PC (a partire dal processore 80486, 16 MByte di memoria principale)

Repertorio linguistico	

Operatori	`Potenza/Aritmetica`
	`Confronti/`
	`Combinazioni`

Funzioni	`Temporizzatori/Contatori/`
	`Richiami di blocchi`
	`funzionali`

Strutture di controllo	`Loop (FOR/WHILE/REPEAT)` `Alternative (IF THEN/CASE/GOTO)`
Tipi di dati	

semplici	`BOOL/BYTE/WORD/DWORD/`
	`INT/DINT/REAL`
	`DATE/TIME/TIME_OF_DAY/S5TIME`

composti	`Stringhe/campi/strutture/strutture` `definiti dall'utente/DATE_AND_TIME`

Sommario del capitolo

1.1 Che cos'è SCL?

Linguaggio di programmazione avanzato

SCL (*Structured Control Language*) è un linguaggio di programmazione avanzato orientato a PASCAL. Il linguaggio si basa su una norma per PLC (=controllori programmabili).

La norma DIN EN-61131-3 (int. IEC 1131-3) normalizza i linguaggi di programmazione per controllori programmabili. La base per SCL è la parte relativa al *"testo strutturato"*. Per informazioni più dettagliate sulla conformità alla norma si rimanda alla "Compliance List" nel file NORM.TAB di STEP 7.

Oltre ad elementi di linguaggi avanzati, SCL contiene anche tipici elementi di PLC come ingressi, uscite, temporizzatori, merker, richiami di blocchi ecc. come elementi del linguaggio. Cioè, SCL integra ed amplia il software di programmazione STEP 7 con i linguaggi di programmazione KOP e AWL.

Ambiente di sviluppo

Per un impiego ottimale e pratico di SCL esiste un efficiente ambiente di sviluppo che tiene conto delle specifiche esigenze sia di SCL che di STEP 7. Questo ambiente di sviluppo si compone di:

- un **Editor**, per programmare programmi composti di funzioni (FC), blocchi funzionali (FB), blocco organizzativi (OB), blocchi dati (DB) e tipi di dati definiti dall'utente (UDT). Facilitano il lavoro del programmatore numerose funzioni efficaci.
- un **Batch-Compiler** per compilare ossia convertire in codice macchina MC7 il programma editato in precedenza. Il codice MC7 generato può essere eseguito su tutte le CPU del sistema di automazione S7-300/400 a partire dalla CPU 314.
- un **Debugger**, che consente la ricerca di errori logici di programmazione in una compilazione esente da errori. La ricerca degli errori avviene nel linguaggio sorgente.

I singoli componenti sono di uso facile e confortevole, poiché essi operano in ambiente Windows 95 e sfruttano così tutti i vantaggi offerti da questo sistema operativo.

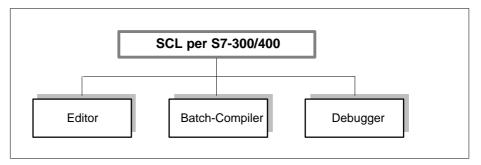



Figura 1-1 Ambiente di sviluppo di SCL

1.2 Quali vantaggi offre SCL all'utente?

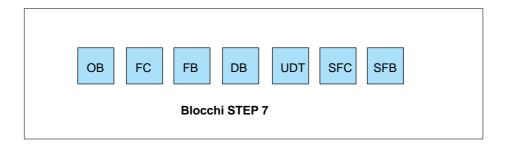
Linguaggio di programmazione avanzato

SCL offre tutti i vantaggi di un linguaggio di programmazione avanzato. SCL possiede però anche alcune proprietà concepite appositamente per supportare la tecnica di programmazione strutturata, p. es.:

- la struttura a blocchi di STEP 7
- blocchi predefiniti
- compatibilità con STEP 5

Struttura a blocchi STEP 7 di provata efficacia pratica

SCL è stato concepito in modo particolare per risolvere qualsiasi tipo di problema che può verificarsi nei progetti di automazione, per consentire all'utente di operare in STEP 7 con la massima efficacia in tutte le fasi del progetto.

SCL supporta in modo particolare il concetto dei blocchi di STEP 7 e consente perciò, oltre a AWL e KOP, la programmazione di blocchi conformemente alle norme specifiche.

Tipi di blocchi

Per funzione, struttura e scopo, i blocchi STEP 7 sono parti ben definite di un programma utente. Con SCL si possono realizzare i blocchi seguenti:

Abbreviazione	Tipo di blocco	Funzione
OB	Blocco organizzativo	Interfaccia fra sistema operativo e programma utente.
FC	Funzione	Blocco con possibilità di trasferimento di parametri senza memoria.
FB	Blocco funzionale	Blocco con possibilità di trasferimento di parametri con memoria.
DB	Blocco dati	Blocco per il deposito di dati utente.
UDT	Tipo di dati definito dall'utente	Blocco per il deposito di un tipo di dati definito dall'utente

Blocchi predefiniti

Non è necessario programmare ogni singola funzione. L’utente può anche accedere a blocchi predefiniti. Questi ultimi sono presenti nel sistema operativo dell’unità centrale o nelle biblioteche *(S7lib)* del pacchetto STEP 7 e possono essere utilizzati p. es. per programmare funzioni di comunicazione. Si tratta in particolare dei seguenti tipi di blocchi:

Abbreviazione	Tipo di blocco	Funzione
SFC	Funzione di sistema	Proprietà analoghe a quelle di una funzione (FC)
SFB	Blocco funzionale di sistema	Proprietà analoghe a quelle di un blocco funzionale (FB)

Possibilità di ”combinare” i blocchi

I blocchi programmati con SCL possono essere ”combinati” con blocchi AWL, KOP e FUP. Il altre parole, un blocco programmato con SCL può richiamare un altro blocco programmato in AWL, KOP o FUP. Analogamente, i blocchi SCL possono essere richiamati in programmi AWL, KOP e FUP. Perciò, i linguaggi di programmazione di STEP 7 e SCL (come pacchetto opzionale) si integrano a vicenda in modo ottimale.

Possibilità di ricompilazione

Nelle applicazioni pratiche, i blocchi SCL possono essere ricompilati nel linguaggio di programmazione STEP 7 AWL (lista istruzioni). Non è invece possibile la ricompilazione verso SCL.

Compatibilità con STEP 5

I blocchi che sono stati programmati con SCL in STEP 5, tranne alcune eccezioni, sono compatibili verso l’alto, cioè questi blocchi possono essere editati, compilati e testati anche con SCL per STEP 7.

Metodi di programmazione

SCL supporta la programmazione strutturata mediante moderne tecniche di Software engineering.

Capacità di apprendimento

Se si possiede una certa esperienza in un linguaggio di programmazione avanzato, SCL può essere appresa facilmente, poiché il repertorio degli elementi linguistici disponibili in SCL si orienta a linguaggi di programmazione avanzati.

1.3 Caratteristiche funzionali dell'ambiente di sviluppo

Editor

L'Editor SCL è un editor di testi, che consente l'elaborazione di qualsiasi tipo di testo. Il compito principale realizzabile con un editor è la generazione e l'elaborazione di file sorgente per programmi STEP 7. In un file sorgente si possono programmare uno o più blocchi:

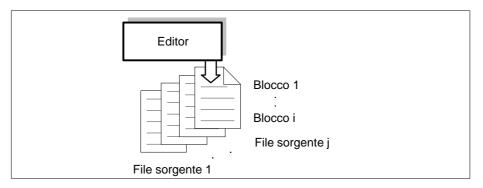



Figura 1-2 Editor SCL

L'editor SCL consente:

- di editare un intero file sorgente con uno o più blocchi.
- di editare un file di comando compilazione che consente di automatizzare la compilazione di molti file sorgente.
- di utilizzare funzioni supplementari, che consentono una pratica elaborazione del testo sorgente, p. es. Cerca/Sostituisci.
- di impostare in modo ottimale l'editor in base alle proprie esigenze specifiche.

Durante l'immissione dati non viene eseguito alcun controllo sintattico.

Compilatore

Dopo aver creato i propri file sorgente con l'Editor SCL, essi devono essere compilati ossia convertiti in codice MC7.

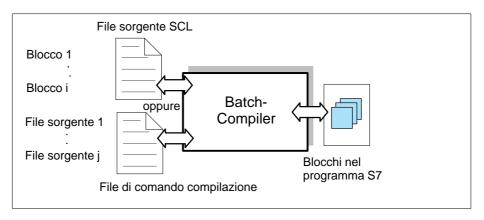



Figura 1-3 Compilatore SCL

Il compilatore SCL consente:

- di compilare un file sorgente SCL con più blocchi in un solo lancio di compilazione.
- di compilare più sorgenti SCL mediante un file di comando compilazione contenente i nomi dei file sorgente.
- di impostare in modo ottimale il compilatore in base alle proprie esigenze specifiche.
- di visualizzare tutti gli errori e i messaggi di avviso che si verificano durante la compilazione.
- di localizzare con facilità i passaggi del testo sorgente contenente errori, eventualmente, come opzione, con descrizione dell'errore e indicazioni sul modo di eliminare l'errore.

Debugger

Il debugger SCL consente di controllare lo svolgimento di un programma nel PLC, per poter identificare eventuali errori logici.

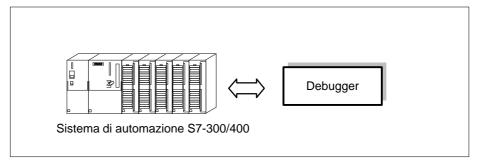



Figura 1-4 Debugger SCL

SCL offre in tal senso due diversi modi di test:

- l'osservazione **passo passo**. Viene controllato l'intero svolgimento logico del programma. L'algoritmo del programma può essere eseguito istruzione per istruzione e in una finestra risultati si può osservare come vengono modificate le variabili elaborate.
- l'osservazione **continua**. Si può testare un intero gruppo di istruzioni nell'ambito di un blocco della sorgente. Durante lo svolgimento del test, i valori delle variabili ed i parametri vengono visualizzati in sequenza cronologica e – se possibile – aggiornati ciclicamente.

Pacchetto STEP 7

L'ambiente di sviluppo SCL offre la possibilità di richiamare le funzioni del pacchetto STEP 7 come per esempio visualizzare e cambiare lo stato di funzionamento della CPU ed impostare l'ora direttamente in SCL.

Sviluppo di programmi SCL

2

Introduzione

Come è dimostrato nella pratica, il modo più semplice e veloce di programmare viene realizzato strutturando i compiti da eseguire, ovvero scindendoli in parti autonome. In questo l'utente viene supportato da SCL, il quale consente di sviluppare singoli blocchi in modo razionale ed efficace.

Questo capitolo descrive il modo in cui sviluppare e implementare un programma utente in SCL. La descrizione viene facilitata da un programma esempio, il quale potrà essere eseguito con i dati di test forniti e con le unità d'ingresso/uscita dell'utente.

Sommario del capitolo

2.1 Panoramica

Obiettivo

La parte inerente lo sviluppo di programmi, intende insegnare all’utente come impiegare SCL nel modo più efficace. Le domande più frequenti all’inizio possono essere:

- In che modo posso procedere per creare un programma in SCL?
- Quali mezzi linguistici offre SCL per la soluzione del problema?
- Di quali funzioni di test posso disporre?

Queste ed altre domande vengono trattati in questo capitolo.

Mezzi del linguaggio SCL

Nell’esempio riportato vengono descritti fra l’altro i seguenti elementi del linguaggio SCL:

- Struttura e impiego dei vari tipi di blocchi SCL
- Richiamo dei blocchi con trasferimento e analisi dei parametri
- Vari formati d’ingresso/uscita
- Programmazione con tipi di dati semplici e campi
- Inizializzazione delle variabili
- Strutture del programma con impiego di diramazioni e loop

Requisiti hardware

Il programma esempio può essere eseguito con SIMATIC S7-300 o SIMATIC S7-400; a tal fine è necessaria la seguente periferia:

- un’unità d’ingresso con 16 canali
- un’unità di uscita con 16 canali

Funzioni di test

Il programma è concepito in modo tale da poter essere rapidamente testato tramite gli interruttori sull’unità d’ingresso e gli elementi di visualizzazione sull’unità di uscita. Per l’esecuzione di un test dettagliato, utilizzare le funzioni di test di SCL, vedere capitolo 6.

Oltre ai vantaggi del linguaggio, l’utente ha a disposizione di tutte le funzioni che offre il pacchetto STEP 7.

2.2 Descrizione del problema

Panoramica

I valori di misura devono essere rilevati, ordinati ed elaborati tramite l'unità d'ingresso. Il campo dei valori di misura va da 0 a 255. È pertanto necessario per l'introduzione un byte.

Come funzioni di elaborazione si devono utilizzare le funzioni di radice e quadrato. I risultati devono essere visualizzati tramite un'unità di uscita, il che richiede una parola. Il comando del programma avviene mediante un byte d'ingresso.

Rilevazione dei valori di misura

Un valore di misura, impostato tramite 8 interruttori d'ingresso, deve essere trasferito nella memoria del campo valori di misura quando sull'interruttore d'ingresso viene riconosciuto un fronte – vedere rappresentazione alla figura 2-1. Il campo dei valori di misura deve essere organizzato come buffer circolare con massimalmente 8 registrazioni.

Elaborazione dei valori di misura

Quando sull'interruttore di ordinamento viene riconosciuto un fronte, i valori memorizzati nel campo valori di misura devono essere ordinati in ordine ascendente. Quindi, per ciascun valore si devono calcolare la radice e il quadrato.

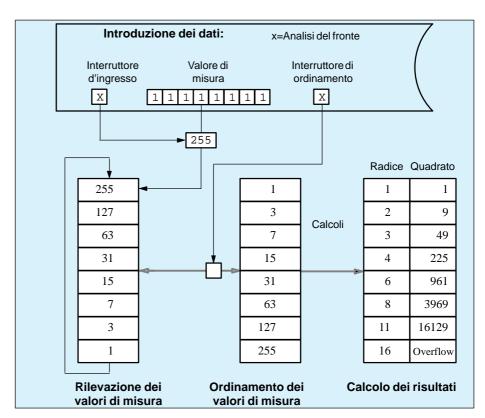



Figura 2-1 Rilevazione ed elaborazione dei valori di misura

Uscita impostabile

Poiché all'uscita viene visualizzato solo un valore per volta, vi devono essere le seguenti possibilità di scelta:

- Scelta di un elemento all'interno di una lista
- Scelta fra valore di misura, radice e quadrato

La scelta di un elemento all'interno di una lista deve avvenire in modo che un elemento della lista venga indirizzato tramite la seguente impostazione mediante interruttori:

- Tramite tre interruttori viene impostata una codifica che viene confermata se sul quarto interruttore, l'interruttore di codifica, viene riconosciuto un fronte. Viene quindi calcolato l'indirizzo usato per indirizzare l'uscita.
- Con lo stesso indirizzo vengono preparati per l'uscita tre valori: valore di misura, radice e quadrato. Per poter scegliere uno dei tre valori, si devono predisporre – come rappresentato alla figura 2-2 – due commutatori.

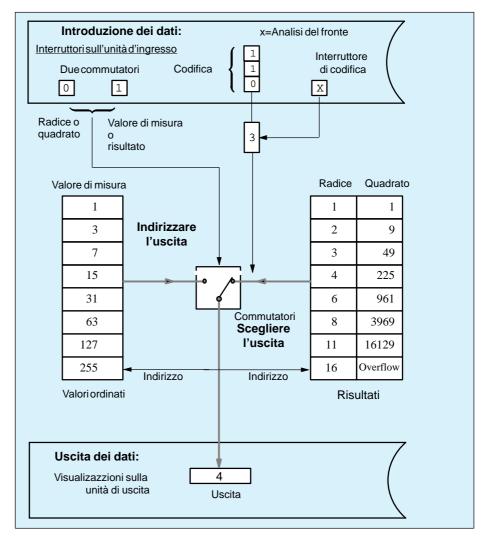



Figura 2-2 Esempio parte 2: uscita impostabile

2.3 Soluzione con blocchi SCL

Panoramica

Il modo migliore per risolvere il problema descritto è quello di utilizzare un **programma SCL strutturato**. Questo programma ha una struttura modulare, cioè è suddiviso in blocchi che eseguono uno o piu compiti parziali. In SCL, così come negli altri linguaggi di programmazione di STEP 7, l'utente ha la scelta tra numerosi tipi di blocchi. Per informazioni in proposito, consultare i capitolo 1, 7 e 8).

Guida alla soluzione

L'approccio alla soluzione può aver luogo con i passi seguenti:

1. Elenco dei compiti parziali
2. Scelta e attribuzione dei blocchi ai compiti parziali
3. Definizione delle interfacce fra i blocchi
4. Definizione dell'interfaccia d'ingresso/uscita
5. Programmazione dei blocchi

2.3.1 Definizione dei compiti parziali

Panoramica

I compiti parziali sono illustrati con caselle nella figura 2-3. Le aree quadrate su sfondo grigio rappresentano i blocchi. La disposizione dei blocchi di codice da sinistra verso destra corrisponde alla sequenza di richiamo.

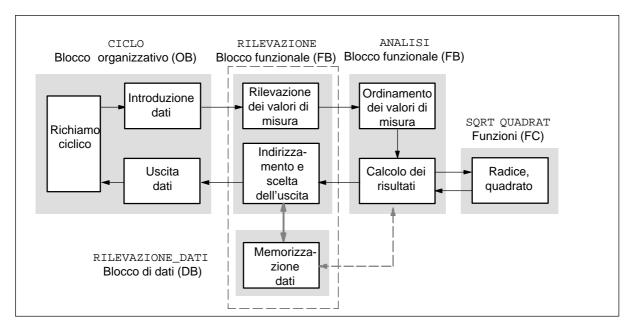



Figura 2-3 Formazione di blocchi di dati dai compiti parziali

2.3.2 Scelta e attribuzione dei blocchi dati ai compiti parziali

Panoramica

In seguito vengono descritti i criteri con i quali è stata eseguita la scelta dei blocchi:

`CICLO`

I programmi utente possono essere avviati solo in OB. Poiché i valori devono essere presenti e rilevati in continuazione, è necessario un OB per il *richiamo ciclico* (OB1). Una parte dell'elaborazione viene programmata nell'OB: *Introduzione dati* e *Uscita dati*.

`RILEVAZIONE`

Per il compito parziale *Rilevazione valori di misura* è necessario un blocco con memoria cioè un FB, poiché determinati dati dei blocchi locali (p. es. il buffer circolare) devono essere conservati da un ciclo di programma all'altro. Il luogo per la *Memorizzazione dati*, detto anche memoria, è il blocco dati di istanza `RILEVZIONE_DATI`.

Lo stesso FB può anche eseguire il compito parziale *Indirizzamento e scelta dell'uscita*, poiché esso dispone dei dati necessari.

`ANALISI`

Per la scelta del tipo di blocco per risolvere i compiti parziali *Ordinamento dei valori di misura* e *Calcolo dei risultati* si deve tener presente che è necessario creare un buffer d'uscita, il quale per ogni valore di misura contiene i risultati dei calcoli della radice e del quadrato.

Perciò, come blocco si può utilizzare solo un FB. Poiché l'FB viene richiamato da un FB sovraordinato, esso non ha bisogno di un DB proprio. I suoi dati di istanza possono essere depositati nel blocco dati di istanza dell'FB richiamante.

`SQRT (Radice) e QUADRAT`

Per risolvere il compito parziale *Calcolo della radice o del quadrato* la soluzione migliore è una FC poiché il ritorno del risultato può aver luogo come valore della funzione. Inoltre, per il calcolo non sono necessari dati che devono essere conservati più a lungo di un ciclo di elaborazione del programma.

Per il calcolo della radice si può utilizzare `SQRT`, la funzione standard in SCL. Per il calcolo del quadrato si deve creare una funzione `QUADRAT`, la quale deve eseguire anche un controllo dei valori limite del campo di valori.

2.3.3 Definizione delle interfacce fra i blocchi

Panoramica

L'interfaccia di un blocco viene definita tramite la dichiarazione dei suoi **parametri formali**. In SCL esistono le seguenti sezioni di dichiarazione:

- Parametri d'ingresso: dichiarazione con VAR_INPUT
- Parametri di uscita: dichiarazione con VAR_OUTPUT
- Parametri di transito: dichiarazione con VAR_IN_OUT

Quando viene richiamato un blocco, gli vengono trasferiti i dati d'ingresso come **parametri attuali**. Dopo il ritorno al blocco richiamante, i dati di uscita vengono preparati per la conferma. Una FC può trasferire il suo risultato come **valore della funzione**. (Per ulteriori informazioni vedere il capitolo 16.)

RILEVAZIONE

L'OB CICLO non possiede propri parametri formali. Esso richiama l'FB RILEVAZIONE e gli trasferisce il valore di misura e i dati di comando nei suoi parametri formali (tabella 2-1):

Tabella 2-1 Parametri formali di RILEVAZIONE

Nome	Tipo di dati	Tipo di dichiarazione	Descrizione
valoremi-sura_inse-rito	INT	VAR_INPUT	Valore di misura
valorenuovo	BOOL	VAR_INPUT	Interruttore per confermare il valore di misura nel buffer circolare
ordinamento-nuovo	BOOL	VAR_INPUT	Interruttore per confermare il valore di misura nel buffer circolare
sceltafun-zione	BOOL	VAR_INPUT	Commutatore per scegliere radice o quadrato
scelta	WORD	VAR_INPUT	Codifica per scegliere il valore di uscita
sceltanuova	BOOL	VAR_INPUT	Interruttore per confermare la codifica
risul-tato_uscita	DWORD	VAR_OUTPUT	Uscita del risultato calcolato
valoremi-sura_uscita	DWORD	VAR_OUTPUT	Uscita del corrispondente valore di misura

ANALISI

L'FB RILEVAZIONE richiama l'FB ANALISI. Entrambi hanno come dati in comune il campo valori di misura da ordinare. Perciò, esso viene dichiarato come parametro di transito. Per i risultati del calcolo della radice e del quadrato viene creato un campo strutturato come parametro di uscita. Per i parametri formali vedere la tabella 2-2:

Tabella 2-2 Parametri formali di ANALISI

Nome	Tipo di dati	Tipo di dichiarazione	Descrizione
bufferordi-namento	ARRAY[..] OF REAL	VAR_IN_OUT	Campo valori di misura, corrisponde al buffer circolare
buffercal-colo	ARRAY[..] OF STRUCT	VAR_OUTPUT	Campo per i risultati: Struttura con i componenti "ra-dice" e "quadrato" del tipo INT

SQRT end QUADRAT

Le funzioni vengono richiamate da ANALISI. Esse hanno bisogno di un valore d'ingresso e forniscono il loro risultato come valore della funzione, vedere tabella 2-3.

Tabella 2-3 Parametri formali e valori delle funzioni di SQRT e QUADRAT

Nome	Tipo di dati	Tipo di dichiara-zione	Descrizione
valore	REAL	VAR_INPUT	Ingresso per SQRT
SQRT	REAL	Valore della funzione	Radice del valore d'ingresso
valore	INT	VAR_INPUT	Ingresso per QUADRAT
QUADRAT	INT	Valore della funzione	Quadrato del valore d'ingresso

2.3.4 Definizione dell'interfaccia d'ingresso/uscita

Panoramica

La figura 2-4 illustra l'interfaccia d'ingresso/uscita. Si prega di tener presente che, per l'ingresso/uscita byte per byte il byte superiore ha valore minimo e il byte inferiore ha valore massimo. Si ha invece il contrario per l'ingresso/uscita parola per parola:

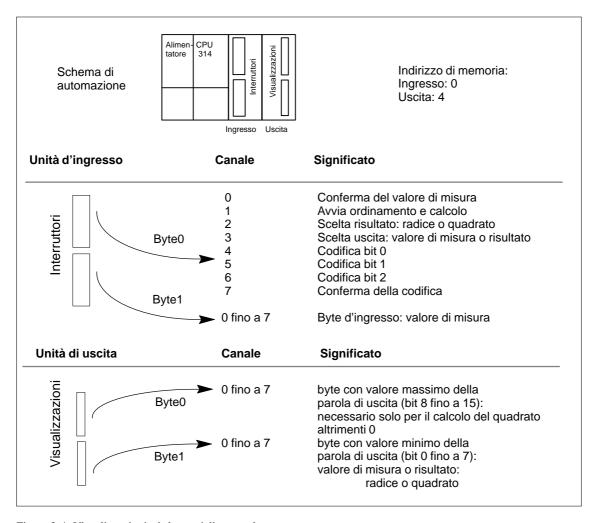



Figura 2-4 Visualizzazioni ed elementi di comando

2.3.5 Programmazione dei blocchi

Programmazione dei blocchi

Una volta definite le interfacce, si possono progettare i blocchi indipendentemente l'uno dall'altro. Il modo migliore è l'approccio "top down", ossia nella sequenza `CICLO, RILEVAZIONE, ANALISI` e `QUADRATO`. In seguito viene fornita una descrizione dei blocchi nella sequenza suddetta.

Quando si compilano dei blocchi si deve tener presente che per poter utilizzare un blocco, cioè prima di poterlo richiamare da un altro blocco, il blocco deve esistere. Ne risulta la sequenza dei blocchi nella sorgente SCL: `QUADRATO, ANALISI, RILEVAZIONE` e `CICLO` (vedere il capitolo 8).

Programmazione simbolica

La comprensione di un programma migliora notevolmente se per gli indirizzi dei blocchi e per i blocchi vengono assegnati **nomi simbolici**. A tal fine è necessario trascrivere le relative registrazioni nella tabella dei simboli come illustra la figura 2-5 (vedere capitolo 7). I nomi possono essere formati in base alla convenzione sui nomi per IDENTIFICATORI o in base alla convenzione per le identificazioni dei segnali (p. es. "Ingresso 0.0"), vedere appendice A.

Commento introduttivo con tabella dei simboli

La figura 2-5 illustra il commento introduttivo alla sorgente SCL con la tabella dei simboli, in cui vengono stabiliti i nomi simbolici, presupposto indispensabile per la compilazione delle sorgente.

```
(*################################################################################

Programma SCL per la rilevazione e l'ulteriore elaborazione di valori di misura:

      –  Tramite l'ingresso 0.0 (interruttore d'ingresso) viene confermato un
         valore di misura presente sull'unità d'ingresso.
      –  L'ulteriore elaborazione dei valori di misura può essere comandata
         tramite diversi interruttori.
      –  Tutti i valori vengono memorizzati nell'area di lavoro del blocco
         funzionale RILEVAZIONE, il blocco di istanza RILEVAZIONE_DATI.

Il programma è stato programmato in modo simbolico. Prima che possa essere compilato, deve
esere presente l'attribuzione dei nomi simbolici agli indirizzi dell'unità e ai blocchi che
vengono eseguiti nella CPU. A tal fine è necessaria la
seguente tabelle dei simboli:

Introduzione                EB1    BYTE // Valore di misura
Ingresso 0.0                E0.0   BOOL // Interruttore d'ingresso per l'immissione
                                           del valore di misura.
Interruttore di ordinamento  E0.1   BOOL // Avvio ordinamento e calcolo
Interruttore di funzione    E0.2   BOOL // Scelta di risultato radice o quadrato
Interruttore di uscita      E0.3   BOOL // Scelta di uscita valore di misura o risultato
Codifica                    EW0    WORD // Codifica, bit significativi 12,13 e 14
Interruttore di codifica    E0.7   BOOL // Conferma della codifica
Uscita                      AW4    INT  // Valore di misura o risultato: radice o quadrato

RILEVAZIONE                 FB10   FB10 // Rilevazione dei valori di misura,
                                        // Indirizzamento e scelta dell'uscita
RILEVAZIONE_DATI            DB10   FB10 // Blocco dati di istanza per RILEVAZIONE
ANALISI                     FB20   FB20 // Analisi dei valori di misura,
                                           calcolo dei risultati
QUADRATO                    FC41   FC41 // Funzione per il calcolo del quadrato
CICLO                       OB1    OB1  // Richiamo ciclico per ingresso/uscita

################################################################################*)
```

Figura 2-5 Commento introduttivo con tabella dei simboli

2.4 Creazione del blocco organizzativo CICLO

Guida alla soluzione

È stato scelto un OB poichè esso viene richiamato **ciclicamente** dal sistema STEP 7. Con esso vengono realizzati i seguenti compiti per il programma:

- Richiamo e alimentazione del blocco funzionale `RILEVAZIONE` con dati d'ingresso e dati di controllo
- Immissione dei dati di risultato del blocco funzionale `RILEVAZIONE`
- Emissione dei valori per la visualizzazione

All'inizio della parte dichiarazioni è presente il campo dati temporaneo con 20 bytes di `"dati di sistema"` (vedere anche il capitolo 8).

```
BLOCCO_ORGANIZZATIVO CICLO
(********************************************************************************
      CICLO corrisponde a OB1, cioè  esso viene richiamato ciclicamente dal
      sistema S7.
      Parte 1  :  Richiamo del blocco funzionale e conferma dei valori d'ingresso
      Parte 2  :  Immissione dei valori di uscita ed emissione con commutazione uscita
********************************************************************************)

VAR_TEMP
            dati di sistema   :     ARRAY[0..20] OF BYTE;   // Area per OB1
END_VAR
BEGIN
(*    Parte 1  :  ***************************************************************)

RILEVAZIONE.RILEVAZIONE_DATI(
      valoremisura_in   := WORD_TO_INT(Ingresso),
      valorenuovo       := "Ingresso 0.0", //Interruttore d'ingresso come simbolo
      ordinamentonuovo  := Interruttore di ordinamento,
      sceltafunzione    := Interruttore di funzione,
      sceltanuova       := Interruttore di codifica,
      scelta            := Codifica);
(*    Parte 2  :  ***************************************************************)

IF Interruttore di uscita THEN                             //Commutazione uscita
            Uscita   := RILEVAZIONE_DATI.risultato_out;     //Radice o quadrato ELSE
            Uscita   := RILEVAZIONE_DATI.valoremisura_out; //Valore di misura END_IF;
END_BLOCCO_ORGANIZZATIVO
```

Figura 2-6 Blocco organizzativo CICLO (OB1)

Conversioni dei tipi di dati

Il valore di misura è presente all'ingresso come tipo BYTE. Esso deve essere convertito in INT. A tal fine esso deve essere convertito da WORD in INT, mentre la precedente conversione da BYTE in WORD avviene automaticamente tramite il compilatore (vedere il capitolo 18). Non è invece necessaria alcuna conversione per l'uscita, poiché quest'ultima è stata dichiarata come INT nella tabella dei simboli, vedere figura 2-5.

2.5 Creazione del blocco funzionale RILEVAZIONE

Guida alla soluzione

Il tipo di blocco FB è stato scelto poiché esistono dati che devono essere salvati da un ciclo di programma all'altro. Si tratta delle variabili statiche che vengono dichiarate nella parte di dichiarazione ”VAR, END_VAR”, vedere tabella 2-4.

Le variabili statiche sono variabili locali i cui valori vengono conservati per tutte le esercuzioni dei blocchi. Esse hanno la funzione di memorizzare i valori di un **blocco funzionale**, e vengono depositate nel blocco dati di istanza.

```
FUNCTION_BLOCK RILEVAZIONE
(*******************************************************************************
      Parte 1 :   Rielevazione dei valori di misura
      Parte 2 :   Avvio ordinamento e calcolo
      Parte 3 :   Analisi codifica e preparazione uscita
*******************************************************************************)
```

Figura 2-7 Intestazione del blocco funzionale RILEVAZIONE

Tabella 2-4 Variabili statiche di RILEVAZIONE

Variabili statiche

Nome	Tipo di dati	Tipo di dichiarazione	Predefinizione	Descrizione
valorimisura	ARRAY [..] OF INT	VAR	8(0)	Buffer circolare per valori di misura
bufferrisul-tati	ARRAY [..] OF STRUCT	VAR	–	Campo per strutture con i componenti ”radice” e ”quadrato” del tipo INT
puntatore	INT	VAR	0	Indice per buffer circolare, registrazione del prossimo valore di misura
valorevec-chio	BOOL	VAR	FALSE	Valore precedente per conferma valore di misura con ”valore nuovo”
ordinamento-vecchio	BOOL	VAR	FALSE	Valore precedente per ordinamento con ”ordinamentonuovo”
sceltavec-chia	BOOL	VAR	FALSE	Valore precedente per conferma della codifica con ”scelta-nuova”
indirizzo	INT	VAR	0	Indirizzo per uscita valore di misura o risultati
anali-si_istanza	ANALISI, = FB 20	VAR	–	Istanza locale per ANALISI FB

Si prega di osservare i valori di impostazione, i quali vengono registrati nelle variabili durante l'inizializzazione del blocco (dopo il caricamento nella CPU). Nella parte di dichiarazione "VAR, END_VAR" viene dichiarata anche l'istanza locale per ANALISI FB. Il nome viene utilizzato in seguito per il richiamo e per i parametri di uscita. Come memoria dati viene utilizzata l'istanza globale RILEVAZIONE_DATI.

Parte dichiarazioni di RILEVAZIONE

In questo blocco, la parte dichiarazioni si compone dei seguenti elementi:

- Dichiarazione delle costanti: fra CONST e END_CONST
- Parametri d'ingresso: fra VAR_INPUT e END_VAR
- Parametri d'uscita: fra VAR_OUTPUT e END_VAR
- Variabili statiche: fra VAR e END_VAR
  Fra queste vi è anche la dichiarazione dell'istanza locale per il blocco ANALISI.

```
CONST
            LIMITE      := 7;
            NUMERO      := LIMITE + 1;
END_CONST

VAR_INPUT
      valoremisura_in      :  INT;  //    Nuovo valore di misura
      valorenuovo          :  BOOL; //    Conferma valore di misura nel  buffer
                                          circolare "valorimisura"
      ordinamentonuovo     :  BOOL; //    Ordinamento dei valori di misura
         sceltafunzione    :  BOOL; //    Scelta della funzione di calcolo
                                          Radice/Quadrato
         sceltanuova       :  BOOL; //    Immissione indirizzo di uscita
            scelta         :  WORD; //    Indirizzo di uscita
END_VAR

VAR_OUTPUT
      risultato_out        :  INT;  //    Valore calcolato
      valoremisura_out     :  INT;  //    Corrispondente valore di misura
END_VAR

VAR
      valorimisura         :  ARRAY[0..LIMITE] OF INT := 8(0);
      bufferrisultato      :  ARRAY[0..LIMITE] OF
         STRUCT
               radice      :  INT;
               quadrato    :  INT;
         END_STRUCT;
      puntatore            :  INT   :=    0;
      valore vecchio       :  BOOL  :=    TRUE;
      ordinamentovecchio   :  BOOL  :=    TRUE;
      sceltavecchia        :  BOOL  :=    TRUE;
      indirizzo            :  INT   :=    0;    //Indirizzo di uscita convertito
      istanza_analisi:  ANALISI;                //Dichiarazione istanza locale
END_VAR
```

Figura 2-8 Parte dichiarazioni del blocco funzionale RILEVAZIONE

Parte istruzioni

La parte istruzioni si compone di 3 parti:

Rilevazione valori di misura

Se il parametro d'ingresso "valorenuovo" è stato modificato rispetto al "valorevecchio", nel buffer circolare viene letto un nuovo valore di misura.

Avvio ordinamento e calcolo

Mediante richiamo della funzione `ANALISI`, se il parametro d'ingresso "ordinamentonuovo" è stato modificato rispetto a "ordinamentovecchio".

Analisi codifica e preparazione uscita

La codifica viene letta parola per parola: conformemente alle convenzioni SIMATIC, ciò significa che il gruppo interruttori superiore (Byte0) contiene gli 8 bit con valore massimo della parola d'ingresso, mentre il gruppo interruttori inferiore (Byte1) contiene i bit con valore minimo. La figura 2-9 illustra le posizioni in cui si trovano gli interruttori tramite i quali si regola la codifica:

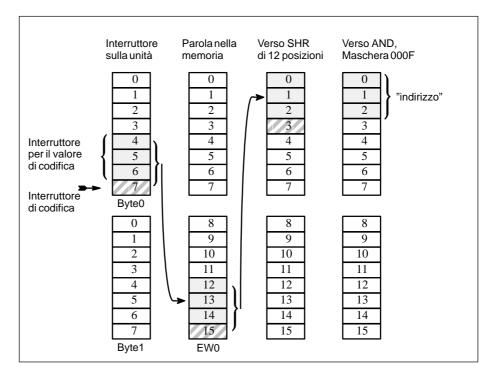



Figura 2-9 Analisi della codifica

Inoltre, la figura 2-9 illustra il calcolo dell'indirizzo: nei bit 12 fino a 14, la parola d'ingresso EW0 contiene la codifica che viene confermata quando, sull'interruttore di codifica (bit 15), viene riconosciuto un fronte. "indirizzo" viene determinato mediante spostamento verso destra con la funzione standard SHR e mascheramento dei bit rilevanti con una maschera AND.

Con questo indirizzo, gli elementi di campo (risultato del calcolo e relativo valore di misura) vengono scritti nel parametro d'uscita. Se viene emessa la radice o il quadrato dipende da "scelta funzione".

Un fronte sull'interruttore di codifica viene riconosciuto dal fatto che "sceltanuova" è cambiata rispetto a "sceltavecchia".

Diagramma di flusso di RILEVAZIONE

La figura 2-10 illustra l’algoritmo in forma di un diagramma di flusso:

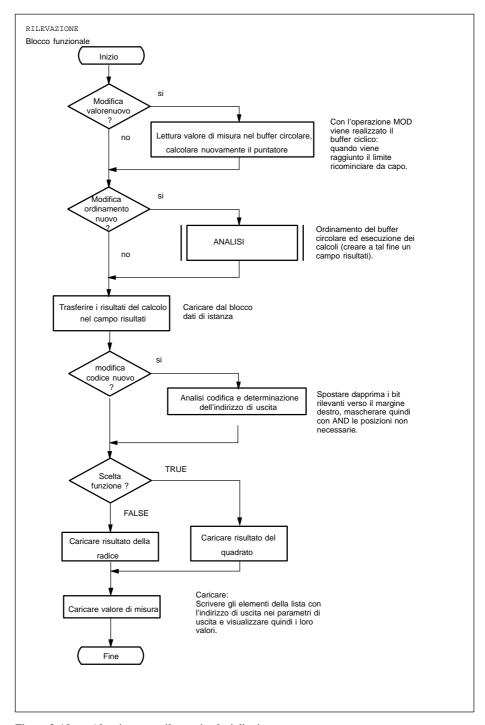



Figura 2-10 Algoritmo per rilevare i valori di misura

Parte istruzioni di RILEVAZIONE

La figura 2-11 illustra la formulazione del diagramma di flusso rappresentato nella figura 2-10 in SCL, cioè la **parte istruzioni** del blocco di codice.

```
BEGIN

(* Parte 1 :   Rilevazione dei valori di misura ***********************************
               In caso di modifica dei ”valorenuovo” viene introdotto il valore di
               misura.
               Con l’operazione MOD si realizza un buffer circolare per i valori di
               misura.*)

IF valorenuovo <> valorevecchio THEN
      puntatore              :=    puntatore MOD ANZAHL;
      valorimisura[puntatore] :=    valoredimisura_in;
      puntatore              :=    puntatore + 1;
END_IF;
valorevecchio  := valorenuovo;

(* Parte 2     :  Avvio ordinamento e calcolo *******************************
                  In caso di modifica di ”ordinamentonuovo” avvio dell’ordinamento
                  del buffer circolare ed esecuzione di calcoli con i valori di
                  misura. I risultati vengono memorizzati in un campo nuovo,
                  ”buffer di calcolo”.*)

IF ordinamentonuovo <> ordinamentovecchio THEN
      puntatore := 0;   //Resettare il puntatore buffer circolare
      istanza_analisi ( bufferordinamento := valorimisura); //Richiamo di ANALISI
END_IF;
ordinamentovecchio   :=    ordinamentonuovo;

bufferrisultati   :=    istanza_analisi.buffercalcolo; //Quadrato e radice

(* Parte 3     :  Analisi codifica e preparazione uscita: ********************
                  In caso di modifica di ”sceltanuova” viene determinata nuovamente
                  la codifica per l’indirizzamento dell’elemento di campo per
                  l’uscita: i bit rilevanti di ”scelta” vengono mascherati e
                  convertiti in integer.
                  A seconda della posizione dell’interruttore ”sceltafunzione”
                  viene preparata ”radice” o ”quadrato” per l’uscita. *)

IF sceltanuova <> sceltavecchia THEN
               indirizzo   := WORD_TO_INT(SHR(IN := scelta, N := 12) AND 16#0007);
END_IF;
sceltavecchia  :=    sceltanuova;

IF sceltafunzione THEN
      risulatto_out  :=    bufferrisultato[indirizzo].quadrato;
ELSE
      risultato_out  :=    bufferrisultato[indirizzo].radice;
END_IF;


valoremisura_out  :=    valorimisura[indirizzo]; //Visualizzazione valori di misura

END_FUNCTION_BLOCK
```

Figura 2-11 Parte istruzioni del blocco funzionale RILEVAZIONE

2.6 Creazione del blocco funzionale ANALISI

Parte dichiarazioni di ANALISI

In questo blocco, la parte dichiarazioni si compone delle seguenti parti:

- Dichiarazione delle costanti: fra CONST e END_CONST
- Parametri di transito: fra VAR_IN_OUT e END_VAR,
- Parametri di uscita: fra VAR_OUTPUT e END_VAR
- Dichiarazione delle variabili temporanee: fra VAR_TEMP e END_VAR

```
FUNCTION_BLOCK ANALISI
(*******************************************************************************
      Parte 1 :      Ordinamento buffer circolare con valori di misura
      Parte 2 :      Avvio calcolo dei risultati
*******************************************************************************)
```

Figura 2-12 Intestazione del blocco funzionale ANALISI

```
CONST
               LIMITE         :=    7;
END_CONST
VAR_IN_OUT
      bufferordinamento       :     ARRAY[0..LIMITE] OF INT;
END_VAR
VAR_OUTPUT
      buffercalcolo           :     ARRAY[0..LIMITE] OF
         STRUCT
            radice            :     INT;
            quadrato          :     INT;
         END_STRUCT;
END_VAR
VAR_TEMP
            scambiare         :     BOOL;
            index, hilf             :     INT;
            valore, risult :  REAL;
 END_VAR
```

Figura 2-13 Parte convenzioni del blocco funzionale ANALISI

Svolgimento

Il parametro di transito ”bufferordinamento” viene associato al buffer circolare ”valorimisura”, cioè il contenuto originale del buffer viene sovrascritto con l’ordinamento dei valori di misura.

Per i risultati di calcolo viene creato il campo nuovo ”buffercalcolo” come parametro di uscita. I suoi elementi sono strutturati in modo da contenere radice e quadrato per ogni valore di misura.

La figura 2-14 descrive il nesso esistente fra i campi descritti.

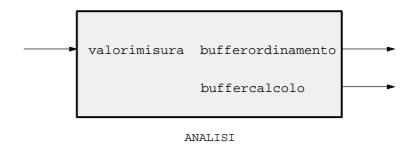



Figura 2-14 Interfaccia dell'FB ANALISI

Questa interfaccia illustra il nucleo dello scambio di dati per l'elaborazione dei valori di misura. I valori vengono memorizzati nel blocco dati di istanza RILEVAZIONE_DATI, poiché nel FB RILEVAZIONE richiamante è stata creata un'istanza locale per l'FB ANALISI.

Parte istruzioni

Dapprima vengono ordinati i valori di misura nel buffer circolare e quindi vengono eseguiti i calcoli:

- Metodo dell'algoritmo per l'ordinamento

Per ordinare il buffer dei valori di misura viene impiegato il metodo di scambio permanente di valori, cioè due valori successivi vengono confrontati fra loro e scambiati fino ad ottenere la sequenza desiderata. Il buffer utilizzato è il parametro di transito "bufferordinamento".

- Avvio del calcolo

Dopo aver completato l'ordinamento, per effettuare i calcoli viene eseguito un loop in cui vengono richiamate la funzione QUADRAT per calcolare il quadrato e la funzione SQRT per calcolare la radice. I risultati ottenuti vengono memorizzati nel campo strutturato "buffercalcolo".

Digramma di flusso di ANALISI

La figura 2-15 illustra l'algoritmo in forma di un diagramma di flusso:

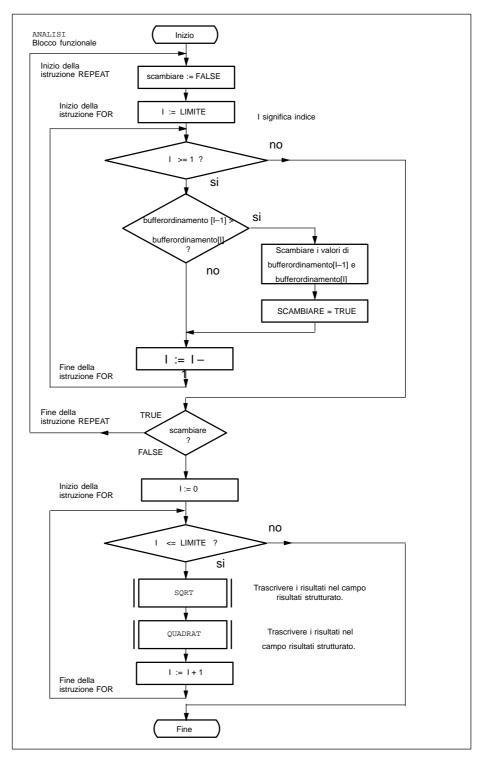



Figura 2-15 Algoritmo per l'analisi dei valori di misura

Parte istruzioni di ANALISI

La figura 2-16 illustra la formulazione del diagramma 2-15 rappresentato nella figura in SCL, cioè la parte istruzioni del blocco di codice:

```
BEGIN

(* Parte 1  Ordinamento : ******************************************************
            in base al metodo ”Bubble Sort”: scambiare in coppia i valori
            fino all’ordinamento del buffer valori di misura. *)

REPEAT
      scambiare      := FALSE;

      FOR index      := LIMITE TO 1 BY –1 DO
         IF bufferordinamento[index–1] > bufferordinamento[index] THEN
            hilf                          :=    bufferordinamento[index];
            bufferordinamento[index]      :=    bufferordinamento[index– 1];
            bufferordinamento[index–1]    :=    hilf;
            scambiare                     :=    TRUE;
         END_IF;
      END_FOR;

UNTIL NOT scambiare
END_REPEAT;


(* Parte 2  Calcolo : ******************************************************
            Calcolo della radice con la funzione standard SQRT e
            calcolo del quadrato con la funzione QUADRAT. *)

FOR index   := 0 TO LIMITE BY 1 DO
      valore         := INT_TO_REAL(bufferordinamento[index]);
      risult         := SQRT(valore);
      buffercalcolo[index].radice        :=    REAL_TO_INT(risult);
      buffercalcolo[index].quadrato      :=    QUADRAT(bufferordinamento[index]);
END_FOR;

END_FUNCTION_BLOCK
```

Figura 2-16 Parte istruzioni del blocco funzionale ANALISI

2.7 Creazione della funzione QUADRAT

Parte istruzioni

Dapprima si verifica se il valore d'ingresso supera il limite massimo consentito per il campo di conteggio di numeri interi. In tal caso, viene trascritto il valore massimo per i numeri interi. Altrimenti viene eseguita la funzione di calcolo del quadrato. Il risultato viene trasferito come valore della funzione.

```
FUNCTION QUADRAT : INT
(*******************************************************************************
Questa funzione fornisce come valore della funzione il quadrato del valore
d'ingresso oppure, in caso di overflow il valore massimo che può essere
rappresentato come numero intero.
*******************************************************************************)
VAR_INPUT
            valore     :   INT;
END_VAR
BEGIN
IF valore <= 181 THEN
            QUADRAT     :=    valore * valore;  //Calcolo del valore della funzione
ELSE
            QUADRAT     :=    32_767;           //in caso di overflow impostare il
                                                //valore massimo
 END_IF;
END_FUNCTION
```

Figura 2-17   Funzione QUADRAT

2.8 Dati di test

Presupposti

Per il test sono necessari un'unità d'ingresso sull'indirizzo 0 e un'unità di uscita sull'indirizzo 4 (vedere figura 2-4).

Prima del test, disporre gli 8 interruttori del gruppo superiore verso sinistra ("0") e gli 8 interruttori del gruppo inferiore verso destra ("1").

Ricaricare i blocchi nella CPU visto che vengono testati anche i valori iniziali delle variabili.

Passi del test

I passi del test sono illustrati nella tabella 2-5.

Tabella 2-5 Passi del test

Test	Azione	Conseguenza
1	Regolare la codifica su "111" (E0.4, E0.5 e E0.6) e confermare con l'interruttore di codifica (E0.7).	Tutte le uscite dell'uscita (byte con valore inferiore) vengono comandate e le visualizzazioni si accendono.
2	Visualizzare la corrispondente radice commutando l'interruttore di uscita (E0.3) su "1".	Le visualizzazioni all'uscita corrispondono al valore binario "10000" (=16).
3	Visualizzare la corrispondente radice commutando l'interruttore di uscita (E0.2) su "1".	Le 15 visualizzazioni all'uscita si accendono. Ciò significa che viene visualizzato un overflow poiché 255 x 255 risulta in un valore troppo grande per il campo integer.
4a	Posizionare l'interruttore di uscita (E0.3) nuovamente su "0".	Viene ora visualizzato nuovamente il valore di misura: tutte le visualizzazioni sulle uscite del byte d'uscita con valore minimo sono impostate.
4b	Come nuovo valore di misura, regolare all'ingresso il valore 3, cioè il valore binario "11".	L'uscita non viene ancora modificata.
5a	Osservare la lettura del valore di misura: regolare la codifica su "000" e confermare con l'interruttore di codifica (E0.7), in modo da poter osservare in seguito l'introduzione dei valori.	All'uscita viene visualizzato 0, cioè nessuna delle visualizzazioni si accende.
5b	Commutare ora l'interruttore d'ingresso "Ingresso 0.0" (E0.0). Ciò causa la lettura del valore regolato nel 4. passo del test.	All'uscita viene visualizzato il valore 3, valore binario "11".
6	Avviare l'ordinamento ed il calcolo mediante commutazione dell'interruzione di ordinamento (E0.1).	All'uscita appare nuovamente 0, poiché in seguito all'operazione di ordinamento il valore di misura è stato ulteriormente spostato verso l'alto.
7	Visualizzare il valore di misura dopo l'ordinamento: regolare la codifica "110" (E0.6 = 1, E0.5 = 1, E0.4 = 0 di EB0, corrisponde a bit 14, bit 13, bit 12 di EW0) e confermare mediante commutazione dell'interruttore di codifica.	All'uscita viene nuovamente visualizzato il valore di misura "11", poiché esso è il secondo valore del campo in ordine di grandezza.
8a	Visualizzare i corrispondenti risultati: mediante commutazione dell'interruttore di uscita (E0.3), viene visualizzato il quadrato del valore di misura del 7. passo del test.	Viene visualizzato il valore d'uscita 9 risp. il valore binario "1001".
8b	Mediante commutazione del tasto funzione (E0.2) si ottiene anche la radice.	Viene visualizzato il valore d'uscita 2 risp. il valore binario "10".

Test supplementare

Le spiegazioni degli interruttori sull'unità di ingresso della tabella 2-6 e gli esempi dei test per quadrato e radice della tabella 2-7 consentono all'utente di definire con facilità i propri passi di test:

- L'introduzione dei dati avviene tramite interruttori: tramite gli 8 interruttori superiori viene eseguito il comando, mentre tramite gli 8 interruttori inferiori viene regolato il valore di misura.
- L'emissione dei dati avviene tramite le visualizzazioni: sul gruppo superiore appare il byte di uscita con valore superiore, mentre sul gruppo inferiore appare il byte con valore inferiore.

Tabella 2-6 Interruttori di comando

Interruttori di comando	Nome	Spiegazione
Canale 0	Interruttore d'ingresso	Commutazione per confermare il valore di misura
Canale 1	Interruttore di ordinamento	Commutazione su ordinamento/analisi
Canale 2	Interruttore di funzione	Interruttore verso sinistra ("0"): radice Interruttore verso destra ("1"): quadrato
Canale 3	Interruttore di uscita	Interruttore verso sinistra ("0"): valore di misura Interruttore verso destra ("1"): risultato
Canale 4	Codifica	Indirizzo di uscita bit 0
Canale 5	Codifica	Indirizzo di uscita bit 1
Canale 6	Codifica	Indirizzo di uscita bit 2
Canale 7	Interruttore di codifica	Commutazione per confermare la codifica

Gli esempi dei test della tabella 2-7 usano tutti gli 8 valori di misura in una sequenza già prestabilita.

Introdurre i valori in una sequenza a piacere. Combinare i relativi bit ed immettere ogni valore mediante commutazione dell'interruttore d'ingresso. Dopo l'introduzione di tutti i valori, avviare l'ordinamento e l'analisi mediante commutazione dell'interruttore di ordinamento. Dopodiché si possono visualizzare i valori di misura ordinati o i risultati – radice o quadrato.

Tabella 2-7 Esempi di test per radice e quadrato

Valore di misura	Radice	Quadrato
0000 0001 = 1	0, 0000 0001 = 1	0000 0000, 0000 0001 = 1
0000 0011 = 3	0, 0000 0010 = 2	0000 0000, 0000 1001 = 9
0000 0111 = 7	0, 0000 0011 = 3	0000 0000, 0011 0001 = 49
0000 1111 = 15	0, 0000 0100 = 4	0000 0000, 1110 0001 = 225
0001 1111 = 31	0, 0000 0110 = 6	0000 0011, 1100 0001 = 961
0011 1111 = 63	0, 0000 1000 = 8	0000 1111, 1000 0001 = 3969
0111 1111 = 127	0, 0000 1011 = 11	0011 1111, 0000 0001 = 16129
1111 1111 = 255	0, 0001 0000 = 16	0111 111, 1111 1111 = Visualizzazione di overflow!

Parte 2: Uso e test

Installazione del software SCL

3

Panoramica

Un programma di Setup consente di installare il software SLC venendo guidati da menu. Il programma di Setup deve essere richiamato con la procedura standard normalmente utilizzata per installare il software sotto Windows 95.

Presupposti per l'installazione

Per poter installare il software SCL sono necessari:

- un dispositivo di programmazione o un PC con il pacchetto base STEP 7 già installato
  - un processore 80486 (o superiore) e
  - una configurazione di memoria RAM di 16 MB
- un monitor a colori, tastiera e mouse supportati da Microsoft Windows 95
- un disco fisso con un spazio di memoria di 78 MB
  (10 MB per dati utente, 60 MB per file Swap, min. 8 MB per il pacchetto opzionale SCL)
- minimo 1 MB di spazio di memoria libera sul drive C: per Setup (i file Setup vengono cancellati dopo aver concluso l'installazione)
- il sistema operativo Windows 95.
- un'interfaccia MPI fra dispositivo di programmazione o PC e PLC:
  - un cavo PC/MPI che viene collegato all'interfaccia di comunicazione del dispositivo dell'utente, oppure
  - un'unità MPI che viene installata nel dispositivo dell'utente. In alcuni dispositivi di programmazione l'interfaccia MPI è già installata.

Sommario del capitolo

3.1 Autorizzazione / licenza d’utilizzo

Introduzione

Per l’uso del pacchetto software SCL è necessaria una specifica autorizzazione (licenza d’utilizzo). Il software così protetto può essere utilizzato solo se sul disco fisso del corrispondente PG/PC viene riconosciuta la necessaria autorizzazione per il programma o il pacchetto software.

Dischetto di autorizzazione

Per l’autorizzazione, l’utente ha bisogno del dischetto di autorizzazione con protezione anticopiatura che fa parte della fornitura. Esso contiene l’autorizzazione ed il programma AUTHORS necessario per visualizzare, installare e disinstallare l’autorizzazione.

Il numero delle possibili autorizzazioni è definito sul dischetto di autorizzazione mediante un contatore di autorizzazioni. Ad ogni autorizzazione, il contatore viene decrementato di 1. Non appena esso raggiunge il valore 0, con questo dischetto non sono più possibili ulteriori autorizzazioni.

Per ulteriori avvertenze e regole per l’uso dell’autorizzazione si rimanda al manuale utente **/231/**.

Attenzione

Osservare le avvertenze contenute nel file LEGGIMI.TXT sul dischetto di autorizzazione. In caso di non osservanza esiste il pericolo di perdere irrevocabilmente l’autorizzazione.

Eseguire la procedura di autorizzazione durante la prima installazione

Si raccomanda di eseguire la procedura di autorizzazione quando l’utente riceve un apposito messaggio di richiesta nell’ambito della prima installazione. Procedere come segue:

1. Inserire il dischetto di autorizzazione quando il sistema emette il corrispondente messaggio di richiesta.
2. Confermare quindi la richiesta.

L’autorizzazione viene trasferita su un drive fisico (cioè, il calcolatore dell’utente ”prende nota” che l’utente è munito di regolare autorizzazione).

Installazione dell’autorizzazione in un momento successivo

Quando un utente avvia il software SCL senza autorizzazione, il sistema visualizza un apposito messaggio. Per eseguire la procedura di autorizzazione, si deve richiamare il programma AUTHORS presente sul dischetto di autorizzazione. Esso consente di visualizzare, installare e disinstallare autorizzazioni. Il programma è guidato da menu.

Avvertenza

Durante l’installazione dell’autorizzazione di SCL, come drive di destinazione si deve indicare il drive C:.

Disinstallazione dell’autorizzazione

Se si rende necessario eseguire una nuova procedura di autorizzazione, p. es. quando si desidera formattare il drive contenente l’autorizzazione, si deve prima ”salvare” l’autorizzazione. A tal fine è necessario il dischetto di autorizzazione originale.

Per trasferire nuovamente l’autorizzazione sul dischetto di autorizzazione procedere come segue:

1. Inserire il dischetto di autorizzazione originale nel drive A: a 3,5”.
2. Richiamare il programma AUTHORS.EXE dal dischetto di autorizzazione.
3. Selezionare il comando di menu **Autorizzazione** ▸ **Disinstalla**.
4. Quindi, nella finestra di dialogo visualizzata introdurre il drive contenente l’autorizzazione e confermare il dialogo. Viene visualizzata una lista di tutte le autorizzazioni contenute nel drive corrispondente.
5. Selezionare l’autorizzazione che si desidera disinstallare e confermare il dialogo. Se l’operazione si conclude senza errori, viene visualizzato il messaggio ”**Autorizzazione** <Nome> **cancellata con successo dal drive** <X:> **.**”
6. Confermare il messaggio.

   Dopodiché, viene visualizzata nuovamente la finestra di dialogo con la lista delle altre autorizzazioni presenti sul drive. Chiudere infine la finestra di dialogo se non si desidera disinstallare ulteriori autorizzazioni.

Il dischetto può in seguito essere nuovamente utilizzato per eseguire la procedura di autorizzazione.

Se il disco fisso è difettoso ...

Se sul disco fisso si verifica un errore prima di poter salvare l’autorizzazione presente su di esso, si prega di contattare la propria rappresentanza SIEMENS.

3.2 Installazione / disinstallazione del software SCL

Panoramica

SCL contiene un programma di Setup che esegue automaticamente l’installazione. Sullo schermo appaiono richieste di introduzione dati che guidano l’utente passo passo nel corso dell’intera installazione.

Preparazioni

Prima di poter iniziare l’installazione, si deve avviare Windows 95 e caricare il pacchetto base STEP 7.

Avvio del programma di installazione

Procedere come segue:

1. Sotto Windows 95 avviare il dialogo per l’installazione del software facendo doppio clic sul simbolo ”Installazione applicazioni” nel ”Pannello di controllo”.
2. Fare clic su ”Installa”.
3. Inserire il supporto dati (dischetto 1) o il CD-ROM e fare clic su ”Continua”. Windows 95 ricerca ora automaticamente il programma di installazione SETUP.EXE.
4. Seguire ora passo passo le istruzioni visualizzate dal programma di installazione.

Il programma guida l’utente attraverso il processo di installazione. L’utente può passare al passo successivo o al passo precedente.

Se è già stata installata una versione SCL

Se il programma di installazione constata che il sistema di origine contiene già un’installazione SCL, viene visualizzato un messaggio corrispondente e l’utente ha le seguenti possibilità di scelta:

- interrompere l’installazione (per poi disinstallare la vecchia versione SCL sotto Windows 95 e avviare quindi la nuova installazione) oppure
- continuare l’installazione e sovrascrivere la vecchia installazione con la nuova versione.

In ogni caso, prima di eseguire un’installazione si consiglia di disinstallare la vecchia versione eventualmente presente nel sistema. La semplice sovrascrizione di una vecchia versione del programma presenta lo svantaggio che, durante la successiva disinstallazione, vi possono essere componenti della vecchia installazione che non vengono rimossi.

Durante il processo di installazione, nella finestra di dialogo vengono visualizzate domande oppure vengono offerte varie opzioni fra le quali l’utente deve scegliere. Si prega di leggere con cura le seguenti note per poter rispondere facilmente e rapidamente alle varie domande rivolte all’utente durante il dialogo.

Disinstallazione

Per eseguire la disinstallazione, si consiglia di adottare il metodo di disinstallazione normalmente usato sotto Windows 95:

1. Sotto Windows 95, avviare il dialogo per l’installazione del software facendo doppio clic sul simbolo ”Installazione applicazioni” nel ”Pannello di controllo”.
2. Selezionare la voce STEP 7 nell’elenco che viene visualizzato relativo ai software installati. Premere quindi il pulsante ”Rimuovi” per disinstallare il software.
3. Se appare la finestra di dialogo ”Are you sure you want to completely remove the selected application and all of its components?”, se vi sono dubbi fare clic sul pulsante ”No”.

Sulla configurazione dell’installazione

Tutti i linguaggi della superficie operativa e tutti gli esempi necessitano di uno spazio di memoria di circa 8 MB.

Sull’autorizzazione

Durante l’installazione viene verificato se sul disco fisso è già presente un’autorizzazione. Se viene riconosciuta un’autorizzazione, appare un messaggio indicante che il software può essere utilizzato solo con la relativa autorizzazione. Se lo si desidera, si può eseguire subito l’autorizzazione oppure continuare l’installazione ed eseguire la procedura di autorizzazione in un secondo tempo.

Nel primo caso, dopo la richiesta del sistema, inserire il dischetto di autorizzazione e confermare. Per ulteriori informazioni sull’autorizzazione vedere il capitolo 3.1.

Conclusione dell’installazione

Se l’installazione viene conclusa con successo, ciò viene visualizzato sullo schermo con un apposito messaggio.

Errori durante l’installazione

I seguenti errori causano l’interruzione dell’installazione:

- Se si verifica un errore di inizializzazione subito dopo l’avviamento di Setup, molto probabilmente il programma SETUP.EXE non è stato avviato sotto Windows 95.
- Lo spazio di memoria non è sufficiente: sul disco fisso vi deve essere uno spazio di memoria libera di almento 8 MB.
- Dischetto difettoso: se si verifica che un dischetto è difettoso, si prega di rivolgersi alla propria rappresentanza Siemens.
- Errore d’uso: iniziare da capo l’installazione e seguire rigorosamente le istruzioni.

4 Come operare con SCL

Panoramica

Questo capitolo contiene informazioni sull'uso di SCL. Esso informa l'utente sulla superficie operativa dell'editor di SCL.

Sommario del capitolo

4.1 Avvio del software SCL

Avvio della superficie operativa Windows

Dopo aver installato il software SCL sul proprio PC, si può avviare SCL anche tramite il pulsante ”Start” sulla barra degli strumenti in Windows 95 (Registrazione sotto ”SIMATIC / STEP 7”).

Avvio dal SIMATIC Manager

Il modo più rapido per avviare SCL è quello di posizionare il puntatore del mouse su una sorgente SCL nel SIMATIC Manager e fare doppio clic. Informazioni al riguardo si trovano nel manuale utente **/231/**.

La figura 4-1 mostra la finestra SCL dopo il richiamo del software.

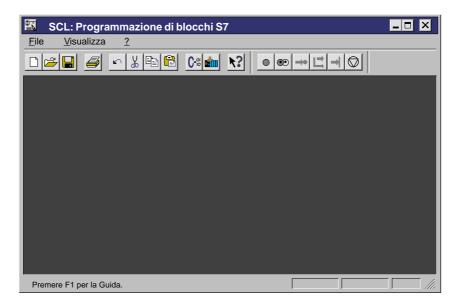



Figura 4-1 Finestra SCL

Avvertenza

Per ulteriori informazioni sull’uso e le operazioni standard di Windows 95 si rimanda al manuale Windows oppure online al programma di esercitazione di Windows 95.

4.2 Adattamento della superficie operativa

Panoramica

Analogamente ad altre finestre STEP 7, le finestre SCL si compongono dei seguenti elementi standard, vedere figura 4-2:

- **Barra del titolo:**
  contiene il titolo della finestra e i simboli per il comando della finestra.
- **Barra dei menu:**
  contiene tutti i menu disponibili nella finestra.
- **Barra degli strumenti:**
  contiene simboli che consentono di eseguire rapidamente i comandi usati più frequentemente.
- **Area di lavoro:**
  contiene le varie finestre nelle quali si può editare il testo del programma oppure leggere informazioni del compilatore e di test.
- **Riga di stato:**
  visualizza lo stato e ulteriori informazioni sull’oggetto selezionato.

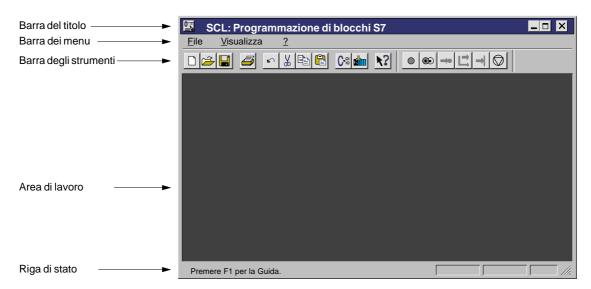



Figura 4-2 Componenti della finestra SCL

Modifica dei componenti

I seguenti componenti possono essere adattati conformemente alle esigenze personali dell'utente:

- Visualizzazione della barra degli strumenti
- Visualizzazione della barra dei punti d'arresto
- Visualizzazione della riga di stato

Adattamento della barra degli strumenti

La visualizzazione della barra degli strumenti può essere inserita o disinserita. A tal fine si deve attivare o disattivare il comando di menu **Visualizza ▸ Barra degli strumenti**. Un segno di spunta dietro il comando di menu indica se il comando è momentaneamente attivato o non.

Adattamento della barra dei punti d'arresto

È possibile attivare o disattivare anche la visualizzazione della barra dei punti d'arresto. A tal fine attivare o disattivare il comando di menu **Visualizza ▸ Barra dei punti d'arresto**. Un segno di spunta dietro il comando di menu indica se il comando è momentaneamente attivato o non.

Adattamento della riga di stato

Anche la visualizzazione della riga di stato può essere attivata o disattivata. A tal fine si deve attivare o disattivare il comando di menu **Visualizza ▸ Riga di stato**. Un segno di spunta dietro il comando di menu indica se il comando è momentaneamente attivato o non.

Adattamento dell'ambiente di sviluppo

L'editor ed il compilatore consentono di eseguire determinate impostazioni atte a facilitare il lavoro all'utente.

- Impostazioni durante la creazione dei blocchi
- Impostazioni per il compilatore
- Impostazioni per l'editor

Creazione dei blocchi

L'utente può ad es. impostare se sovrascrivere durante la compilazione i blocchi già esistenti. Scegliere il comando del menu **Strumenti ▸ Impostazioni** e fare clic nella finestra di dialogo ”Impostazioni” sulla scheda ”Crea blocchi”. Il capitolo 5.5 contiene una descrizione dettagliata delle singole opzioni.

Adattamento del compilatore

Si può adattare anche il processo di compilazione conformemente alle specifiche esigenze dell'utente. Il capitolo 5.5 contiene una descrizione dettagliata delle singole opzioni.

A tal fine si deve scegliere il comando di menu **Strumenti ▸ Impostazioni** e fare clic sulla scheda ”Compilatore” nella finestra di dialogo ”Editor”.

Adattamento dell'editor

Si possono impostare l'ampiezza del tabulatore, il salvataggio prima della compilazione, la visualizzazione dei numeri di riga e altre opzioni. A tal fine si deve scegliere il comando di menu **Strumenti ▸ Impostazioni** e fare clic sulla scheda ”Editor” nella finestra di dialogo ”Impostazioni”.

4.3 Lavorare con l'editor SCL

Panoramica

La sorgente SCL si compone, normalmente, di testo progressivo. Per l'introduzione dei testi, l'editor SCL dispone di funzioni di elaborazione testo concepite appositamente per il sistema SCL.

Finestra dell'editor

L'oggetto sorgente per il programma utente viene introdotto tramite la tastiera nella finestra di lavoro. Si possono aprire diverse finestre di uno stesso oggetto sorgente o di diversi oggetti. La disposizione delle finestre può essere regolata tramite il menu "Finestra".

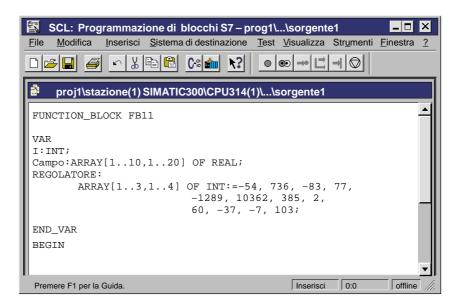



Figura 4-3 Finestra di editazione SCL

Selezione testi

In SCL si può selezionare un testo posizionando il cursore all'inizio dell'area da selezionare e mantenendo premuto il tasto sinistro del mouse mentre si trascina il cursore nell'area desiderata.

Inoltre, si può:

- selezionare l'intero testo di una sorgente scegliendo il comando di menu **Modifica ▸ Seleziona tutto**.
- selezionare una parola facendo doppio clic su di essa, oppure
- selezionare un'intera riga facendo tre volte clic su di essa.

Cerca e sostituisci

Con il comando di menu **Modifica ▸ Cerca /Sostituisci** viene aperto una finestra di dialogo tramite la quale si può cercare una stringa di caratteri e sostituirla con altri oggetti di testo.

Inserimento modelli

L'inserimento di modelli consente un'efficiente programmazione e facilita l'osservanza della sintassi. In SCL si può:

- Inserire modelli per blocchi selezionando il comando di menu **Inserisci ▸ Modello di blocco**.
- Inserire modelli per strutture di controllo selezionando il comando di menu **Inserisci ▸ Struttura di controllo**.

Taglia, Copia, Inserisci, Cancella

Si possono tagliare, copiare, inserire e cancellare oggetti di testo nel modo consueto. I corrispondenti comandi di menu sono contenuti nel menu **Modifica**.

In genere, si possono spostare o copiare oggetti anche mediante ”trascinare e rilasciare” (Drag&Drop).

Vai a

Il comando di menu **Modifica ▸ Vai a...** consente di aprire una finestra di dialogo che permette di posizionare il cursore all'inizio della riga desiderata dopo aver indicato il numero della riga e confermato con ”OK”.

Annulla, Ripristina

Il comando di menu **Modifica ▸ Annulla** consente di annullare un'azione, come ad esempio la cancellazione di una riga.
Il comando di menu **Modifica ▸ Ripristina** consente di ripristinare una azione eseguita precedentemente.

Programmare con SCL

5

Panoramica

La programmazione richiede diverse fasi di lavoro. Il presente capitolo ha l'intento di descrivere tali fasi e di prescrivere l'ordine in cui procedere.

Sommario del capitolo

5.1 Creazione di programmi utente in SCL

Presupposti per la creazione di programmi

Prima di creare un programma con SCL si dovrebbero eseguire quanto segue:

1. Creare un progetto con il SIMATIC Manager.
2. Con il SIMATIC Manager, assegnare ad ogni CPU l'indirizzo di comunicazione in rete.
3. Configurare e parametrizzare le unità centrali e le unità di ingresso/uscita.
4. Se si desidera utilizzare indirizzi simbolici per aree di memoria della CPU oppure per nomi di blocchi, si deve creare una tabella dei simboli in cui vengono memorizzati questi nomi.

Creazione della tabella dei simboli

Se si desidera utilizzare indirizzi simbolici per aree di memoria della CPU oppure per nomi di blocchi, si deve creare una tabella dei simboli in cui vengono memorizzati questi nomi. SCL accede a questa tabella durante la compilazione. La creazione della tabella dei simboli e l'assegnazione dei valori in essa avviene con STEP 7.

L'utente può aprire la tabella con il SIMATIC Manager o direttamente con SCL tramite il comando di menu **Strumenti ▸ Tabella dei simboli**.

Inoltre vi è la possibilità di importare e modificare le tabelle che si trovano in forma di file di testo e che possono esser state create con un editor di testo a piacere. (Per maggiori informazioni, consultare **/231/**).

Come procedere?

Per poter creare un programma utente in SCL, si deve dapprima generare una sorgente SCL. In questa sorgente si possono editare e quindi compilare uno o più blocchi (OB, FB, FC, DB e UDT) con un'esecuzione batch. In seguito alla compilazione della sorgente, i blocchi in essa contenuti vengono trasferiti nel contenitore "Programma utente" (<AP-off>, vedere figura 5-1) dello stesso programma S7 in cui è memorizzata anche la sorgente.

Le operazioni di generazioni e editazione della sorgente SCL possono essere eseguite con l'editor integrato o con un editor standard. Le sorgenti che sono state generate con un editor standard devono essere importate nel progetto con il SIMATIC Manager. Dopodiché, esse possono essere aperte e soggette a ulteriore elaborazione o compilazione.

5.2 Generazione e apertura di una sorgente SCL

Panoramica

Le sorgenti generate con SCL possono essere integrate nella struttura di un programma S7 nel modo seguente:

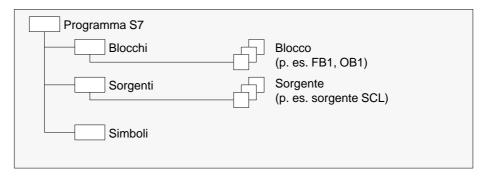



Figura 5-1 Struttura di un programma S7 nel SIMATIC Manager

Generazione della sorgente SCL

Per creare una sorgente SCL si deve procedere come segue:

1. Selezionare il comando di menu **File ▸ Nuovo** o fare clic sul simbolo ”Nuovo” nella barra degli strumenti.
2. Nella finestra di dialogo ”Nuovo”, selezionare il progetto desiderato e il contenitore di sorgenti (SO) nel corrispondente programma S7.
3. Aprire il contenitore sorgente e selezionare il comando di menu **Inserisci ▸ Software S7 ▸ Sorgente SCL**.
4. Selezionare la sorgente e scegliere il comando di menu **Modifica ▸ Proprietà dell'oggetto**. Nella scheda ”Generale” introdurre il nome dell'oggetto sorgente. Il nome può essere lungo max. 24 caratteri. Nei nomi delle sorgenti si tiene conto delle maiuscole e delle minuscole.
5. Fare doppio clic sulla sorgente. Viene aperta una finestra vuota in cui è possibile editare la sorgente SCL.

Apertura di una sorgente SCL

Si può aprire un oggetto sorgente, generato e salvato in precedenza con SCL, per sottoporlo a ulteriore compilazione o elaborazione.

Procedere nel modo seguente:

1. Selezionare il comando di menu **File ▸ Apri...**, o fare clic su simbolo ”Apri”.
2. Nella finestra di dialogo ”Apri”, selezionare il progetto desiderato e il corrispondente programma S7.
3. Assicurarsi che sia impostato il filtro ”Sorgente SCL” e selezionare il contenitore sorgente.
4. Nella finestra di dialogo vengono visualizzate tutte le sorgenti SCL del programma S7. Selezionare l'oggetto desiderato e confermare con ”OK” o fare doppio clic sulla sorgente.

Le sorgenti che sono state generate con un editor standard, possono essere aperte in modo analogo dopo che esse sono state importate nel progetto tramite il SIMATIC Manager.

5.3 Introduzione di dichiarazioni, istruzioni e commenti

Panoramica

L'introduzione di una sorgente SCL deve essere eseguita conformemente a regole sintattiche ben definite. Queste regole fanno parte integrante della *descrizione del linguaggio* e sono completamente elencate in appendice.

proj1\stazione(1)SIMATIC300\CPU314(1)\...\sorgente1

```
FUNCTION_BLOCK FB11

VAR
I:INT;
Campo: ARRAY[1..10,1..20] OF REAL;
REGOLATORE:
       ARRAY[1..3,1..4] OF INT:=–54, 736, –83, 77,
                                 –1289, 10362, 385, 2,
                                 60, –37, –7, 103;

END_VAR
BEGIN
```

Figura 5-2 Sorgente SCL

Regole

L'introduzione deve essere eseguita rispettando le seguenti convenzioni:

- In una sorgente SCL si possono formulare tanti blocchi di codice (FB, FC, OB), blocchi dati (DB) e dati definiti dall'utente (UDT) quanti se ne desiderano, in cui ogni tipo di blocco possiede la sua struttura tipica (vedere capitolo 8)
- I caratteri maiuscoli e minuscoli sono rilevanti solo per gli identificatori simbolici (p. es. per nomi di variabili e literal di caratteri).
- I blocchi richiamati sono situati prima dei blocchi in cui essi vengono richiamati.
- I tipi di dati definiti dall'utente (UDT) sono situati prima dei blocchi in cui essi vengono utilizzati.
- I blocchi dati globali sono situati prima dei blocchi che accedono ai suddetti blocchi.
- Osservare le norme sul formato e sulla sintassi descritte nella parte *Descrizione del linguaggio* del presente manuale.

5.4 Salvataggio e stampa di una sorgente SCL

Salvataggio di una sorgente SCL

Con il termine ”Salvare”, in SCL si intende sempre il salvataggio di sorgenti. In SCL, i blocchi vengono generati durante la compilazione della sorgente e memorizzati automaticamente nella corrispondente directory di programma.

Per salvare una sorgente SCL esistono diverse possibilità:

- Selezionare il comando di menu **File ▸ Salva**, o fare clic sul simbolo ”Salva” nella barra degli strumenti.

  La sorgente SCL viene aggiornata.

- Se si desidera creare una copia della sorgente SCL attuale, selezionare il comando di menu **File ▸ Salva con nome**. Appare la finestra di dialogo ”Salva con nome”, in cui si può indicare il nome e il percorso del duplicato.
- Quando viene eseguito il comando di menu **File ▸ Chiudi** e una sorgente SCL modificata non è stata ancora salvata, il sistema chiede se l’utente desidera salvare o annullare le modifiche apportate oppure se si desidera interrompere il comando **Chiudi**.

  Al posto del comando di menu **File ▸ Chiudi** si può anche fare clic sul simbolo ”Chiudi” nella riga del titolo.

  Anche quando si desidera uscire da SCL tramite il comando di menu **File ▸ Esci**, non avendo ancora salvato lo stato attuale delle sorgenti aperte, per ogni sorgente viene visualizzato un dialogo in cui l’utente può salvare o annullare le modifiche apportate.

Stampa dell’oggetto sorgente

In qualsiasi momento, è possibile stampare blocchi, dichiarazioni e istruzioni presenti nella sorgente SCL. A tal fine si deve dapprima installare e impostare la stampante tramite il pannello di controllo di Windows. Procedere nel modo seguente:

- Fare clic sul simbolo ”Stampa” nella barra degli strumenti oppure selezionare il comando di menu **File ▸ Stampa...**. Viene visualizzata una finestra di dialogo, in cui si possono selezionare diverse opzioni di stampa come p. es. l’area di stampa oppure il numero di copie da stampare.

  Confermare con ”OK” per inviare il documento alla stampante.

Impostazione della pagina

Tramite il comando di menu **File ▸ Imposta pagina** si possono definire il formato della pagina di stampa.

Generazione delle righe di intestazione e di piè di pagina

Le righe di intestazione e di piè di pagina dei documenti da stampare possono essere impostate nel SIMATIC Manager con il comando di menu **File ▸ Campi di scrittura**.

Anteprima di stampa

Tramite il comando di menu **File ▸ Anteprima di stampa** si possono controllare le impostazioni eseguite con il comando ”Imposta pagina” prima di avviare la stampa del documento. Qui non è possibile eseguire alcuna modifica del documento.

5.5 Processo di compilazione

Panoramica

Prima di poter eseguire o testare un programma utente, quest’ultimo deve essere compilato. Il compilatore viene attivato con l’avvio del processo di compilazione (vedi sotto). Il compilatore ha le seguenti caratteristiche:

- Il compilatore lavora nella modalità operativa Batch, cioè esso elabora una sorgente SCL come intera unità. Non sono possibili compilazioni parziali (p. es. riga per riga).
- Il compilatore verifica la sintassi di una sorgente SCL e visualizza tutti gli errori riscontrati nel corso della compilazione.
- Esso genera blocchi con informazioni di test se la sorgente SCL è esente da errori e se è stata impostata l’opzione corrispondente (vedere sotto). L’opzione informazioni di test deve essere selezionata per ogni programma che si desidera testare con SCL a livello di linguaggio avanzato.
- Ad ogni richiamo di un blocco funzionale, esso genera un corrispondente blocco dati di istanza purché quest’ultimo non sia già esistente.

Impostazione del Compiler

Si ha la possibilità di adattare il processo di compilazione in base alle esigenze specifiche dell’utente. A tal fine si deve selezionare il comando di menu **Strumenti ▸ Impostazioni** e nella finestra di dialogo ”Impostazioni” fare clic sulla scheda ”Compilatore”. Le opzioni possono essere inserite e disinserite facendo clic con il mouse.

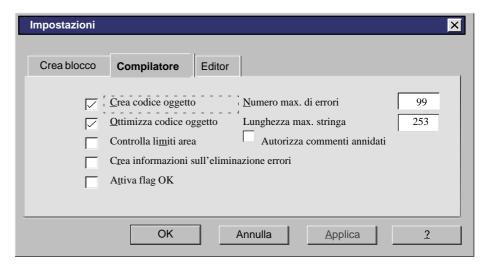



Figura 5-3 Finestra di dialogo ”Impostazioni”, scheda ”Compilatore”

Opzioni

Le opzioni hanno i seguenti significati:

- **Numero massimo di errori**: numero massimo di errori. Il Compiler interrompe la compilazione di una sorgente SCL quando viene raggiunto il numero di errori indicato.
- **Crea codice dell’oggetto**: generare con codice eseguibile su un PLC sì/no
- **Ottimizza codice dell’oggetto**: generare un codice più breve. Una volta selezionata l’opzione delle informazioni sul test non tutte le ottimizzazioni sono possibili.
- **Controlla limiti dell’area:** verificare, durante l’esecuzione, se nell’area consentita per questo campo – secondo la convenzione – sono contenuti indici di campo. Se un indice di campo supera l’area consentita, il flag OK viene impostato su FALSE.
- **Crea informazioni sull’eliminazione degli errori:** generare informazioni di test sì/no. Le informazioni sul test sono necessarie per eseguire i test con il debugger di linguaggi avanzati.
- **Attiva flag OK:** in fase di esecuzione ogni operazione errata dovrebbe impostare la variabile OK su FALSE.
- **Lunghezza massima della stringa:** ridurre la lunghezza standard del tipo di dati ”STRING”. Nell’impostazione di base, la lunghezza standard è di 254 caratteri.
- **Autorizza commenti annidati:** nella sorgente SCL possono essere annidati più commenti l’uno dentro l’altro sì/no.

Creazione del blocco

Nella scheda ”Crea blocco” si hanno varie possibilità di influenzare il processo di compilazione:

- È possibile impostare se durante la compilazione debbano o meno essere sovrascritti blocchi già esistenti.
- È possibile creare dati di riferimento automaticamente durante la compilazione di una sorgente. Se questa opzione è selezionata si allunga tuttavia la procedura di compilazione.
- Fare clic sull’opzione ”Considera attributo di sistema S7_server” se il blocco è rilevante per la progettazione del messaggio e del collegamento. Anche questa opzione allunga la procedura di compilazione.

Avvio del processo di compilazione

Per avviare il processo di compilazione esistono due possibilità:

- selezionare il comando di menu **File ▸ Compila**, oppure
- fare clic sul simbolo ”Compila” nella barra degli strumenti.

Per essere sicuri di avere sempre la versione più recente della sorgente SCL, si raccomanda di selezionare il comando di menu **Strumenti ▸ Impostazioni** e nella scheda ”Editor” di fare clic sull’opzione ”Salva prima di compilare”. In tal modo, il comando di menu **File ▸ Compila** salva implicitamente la sorgente SCL.

Dopo la compilazione

Dopo la compilazione, il sistema segnala una compilazione esente da errori oppure gli errori e i messaggi di avviso vengono visualizzati in una finestra come illustrato nella figura 5-4:

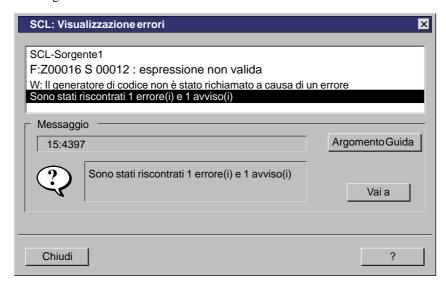



Figura 5-4 Finestra per messaggi di errore e messaggi di avviso

Come trovare le cause degli errori e dei messaggi di avviso

Ciascun messaggio viene indicato con la rispettiva posizione di riga e colonna e una breve descrizione. Per ottenere una dettagliata descrizione dell'errore o del messaggio di avviso si deve selezionare il messaggio desiderato e fare clic sul pulsante ”?”.

Con un doppio clic su un messaggio si può posizionare il cursore sulla corrispondente posizione della sorgente SCL.

Grazie a queste due possibilità, l'utente può localizzare e correggere rapidamente eventuali errori e massaggi di avviso.

5.6 Trasmissione al PLC del programma utente

Panoramica

Durante la compilazione di una sorgente SCL, i blocchi vengono generati dalla sorgente e salvati nel contenitore ”Blocchi” del programma S7. In SCL sono solo questi i blocchi che possono essere caricati in seguito nella CPU dal dispositivo di programmazione.

Se si desidera trasmettere ulteriori blocchi del programma S7 nel controllore programmabile si deve utilizzare il SIMATIC Manager.

Presupposti

Per poter caricare il programma utente nel PLC devono essere soddisfatti i seguenti presupposti:

- Fra il dispositivo di programmazione e il controllore programmabile esiste un collegamento.
- I blocchi che devono essere caricati devono essere stati compilati senza errori.

Cancellazione totale della memoria CPU

La funzione ”Cancellazione totale” consente di cancellare in modo online l’intero programma utente in una CPU. Si deve tener presente che la funzione suddetta resetta la CPU, che tutti i collegamenti esistenti con la CPU vengono interrotti e che, se è inserita una Memory Card, il contenuto della Memory Card viene copiato nella memoria di caricamento interna. Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione ▸ Stato di funzionamento** e commutare la CPU su STOP.
2. Selezionare il comando di menu **Sistema di destinazione ▸ Cancellazione totale**.
3. Confermare l’azione nella finestra di dialogo visualizzata in seguito.

Caricamento nel sistema di destinazione

Il caricamento dei blocchi nello stato di funzionamento STOP presenta determinati vantaggi poiché durante la sovrascrittura di un vecchio programma nello stato di funzionamento RUN si possono verificare degli errori. Procedere nel modo seguente:

1. Selezionare il comando di menu **Sistema di destinazione ▸ Carica**.
2. Se il blocco è già presente nella RAM della CPU, dopo la relativa domanda del sistema si deve confermare l’eventuale sovrascrittura del blocco.

5.7 Creazione di un file di comando compilazione

Panoramica

Si può automatizzare la compilazione di più sorgenti SCL mediante la creazione di un file di comando compilazione.

File di comando compilazione

Per il proprio progetto STEP 7 si può creare un file di comando compilazione. In questo file vengono trascritti i nomi delle sorgenti SCL presenti nel progetto. Queste sorgenti SCL vengono compilate in elaborazione batch.

Creazione

La creazione del file avviene nelle seguenti fasi:

- Quando si genera un file con ”Nuovo” o con ”Apri” si deve creare il filtro per il ”file di comando compilazione”.
- Il file con il quale si lavora adesso ha come contrassegno specifico l’estensione ”.inp”.
- Quando si compila questo file, i file indicati tramite il nome di questo file vengono compilati in successione.

Compilazione

Durante la compilazione, i blocchi generati vengono depositati nel contenitore ”Blocchi” del programma S7.

Test di un programma

6

Panoramica

Le funzioni di test di SCL offrono la possibilità di controllare l'esecuzione di un programma nella CPU e di localizzare in tal modo probabili errori.

Durante la compilazione vengono riconosciuti e visualizzati gli errori sintattici. Anche gli errori del tempo di esecuzione del programma vengono visualizzati tramite allarmi di sistema; con le funzioni di test si possono identificare eventuali errori logici di programmazione.

Dove si trovano ulteriori informazioni?

Per informazioni più dettagliate sul test con SCL si rimanda alla Guida online. Mentre si lavora con SCL, tramite la Guida online, si possono ottenere risposte a domande ben precise.

Sommario del capitolo

6.1 Panoramica

Livello di linguaggio avanzato

Con le funzioni di test di SCL è possibile testare a livello di linguaggio avanzato i programmi utente scritti in SCL. Tramite questo tipo di test si può:

- Scoprire errori di programmazione.
- Osservare e controllare gli effetti di un programma utente sulla struttura della CPU.

Presupposti

Prima di poter testare un programma SCL devono essere soddisfatte le seguenti condizioni:

1. Il programma deve essere stato compilato senza errori con le opzioni ”Crea codice dell’oggetto” e ”Informazioni sul debug”. Si possono selezionare le opzioni nella scheda ”Compilatore” con il comando di menu **Strumenti** ▸ **Impostazioni**.
2. Fra PG/PC e CPU vi deve essere un collegamento online.
3. Inoltre, si deve caricare il programma nella CPU. Ciò avviene tramite il comando di menu **Sistema di destinazione** ▸ **Carica**. I blocchi che non sono stati compilati con SCL devono essere caricati con il SIMATIC Manager.

Funzione di test di SCL

La tabella 6-1 illustra nomi e brevi descrizioni delle principali funzioni di test che possono essere richiamate in SCL.

Tabella 6-1 Una panoramica delle funzioni di test

Funzioni	Descrizione
Controlla continuativamente (CPU S7-300/400)	Emissione di nomi e valori attuali di variabili di un’area di controllo
Punti d’arresto attivi (solo CPU S7-400)	Impostazione, cancellazione ed elaborazione dei punti di arresto: test a passi singoli
Controlla/comanda variabili	Controllo o definizione dei valori attuali dei dati globali
Crea dati di riferimento	Creazione di una panoramica del programma utente
Funzione di test del pacchetto base STEP 7	Interrogazione e modifica dello stato di funzionamento della CPU

Avvertenza

L’esecuzione di un test durante il funzionamento dell’impianto può causare gravi danni a persone e cose in caso di disturbi di funzionamento o errori di programma! Prima di eseguire le funzioni di test ci si deve perciò assicurare che non si possano verificare stati pericolosi!

6.2 Funzione di test ”Controlla continuativamente”

Panoramica

Durante il controllo continuativo di un programma è possibile testare un gruppo di istruzioni. Questo gruppo di istruzioni viene denominato anche area di controllo.

Durante l’esecuzione del test, i valori delle variabili e dei parametri di quest’area vengono visualizzati in sequenza cronologica e aggiornati ciclicamente. Se l’area di controllo si trova in una parte di programma che viene eseguita in ogni ciclo, in genere i valori delle variabili non possono essere rilevati da cicli che si susseguono.

I valori che hanno subito una modifica nella fase attuale di esecuzione vengono visualizzati in nero mentre quelli rimasti invariati vengono visualizzati con caratteri grigio chiaro.

Il volume delle istruzioni da testare dipende dalla potenza di elaborazione della CPU collegata. Poiché le istruzioni SCL del codice sorgente vengono convertite in numerose istruzioni nel programma utente, la lunghezza dell’area di controllo è variabile e viene determinata e selezionata dal debugger SCL quando viene selezionata la prima istruzione dell’area di controllo desiderata.

Modo test

Durante il test con il modo ”Controlla continuativamente” vengono interrogati e visualizzati i valori attuali dei dati dell’area di controllo. L’interrogazione si svolge durante l’elaborazione dell’area di controllo e comporta per lo più un prolungamento dei tempi di ciclo.

Per poter ridurre tale aumento dei tempi di ciclo, SCL offre due diversi ambienti di test.

- **Ambiente di test ”Processo”:**

  Nell’ambiente di test ”Processo”, il debugger SCL limita l’area di controllo massima in modo che i tempi di ciclo durante l’esecuzione del test non superino, o superino solo in modo irrilevante, i tempi reali di esecuzione del processo.

- **Ambiente di test ”Laboratorio”:**

  Nell’ambiente di test ”Laboratorio”, l’area di osservazione viene limitata solo dalla potenza di elaborazione della CPU collegata. Si può però avere un aumento dei tempi di ciclo rispetto al processo reale e l’area di controllo massima è maggiore di quella dell’ambiente di test ”Processo”.

Come si impiega ”Controlla continuativamente”?

Per eseguire la funzione ”Controlla continuativamente” procedere nel modo seguente:

1. Assicurarsi che siano soddisfatti i requisiti descritti nel capitolo 6.1.
2. Selezionare la finestra che contiene la sorgente del programma da testare.
3. Per modificare eventualmente l’ambiente del test preimpostato (processo), selezionare il comando di menu **Test ▸ Ambiente di test ▸ Laboratorio**.
4. Posizionare il cursore sulla riga del testo sorgente che contiene la prima istruzione dell’area da testare.
5. Selezionare il comando di menu **Test ▸ Controlla continuativamente.**

   Risultato: l’area massima di controllo possibile viene determinata e visualizzata con una barra grigia sul margine sinistro della finestra. La finestra si suddivide in due: sulla destra verranno visualizzati riga per riga i nomi e i valori attuali delle variabili contenute nell’area di controllo
6. Selezionare il comando di menu **Test ▸ Controlla continuativamente** per concludere il test.
7. Selezionare il comando di menu **Test ▸ Concludi test** per concludere il test.

Avvertenza

Per poter testare un ulteriore blocco nella stessa sorgente non è sufficiente chiudere ”Controlla continuativamente”. Si deve chiudere anche la finestra risultati.

6.3 Funzione di test ”Punti d’arresto attivi”

Panoramica

Con la funzione ”Punti d’arresto attivi” il test viene eseguito in singoli passi. L’utente può eseguire il programma istruzione dopo istruzione e controllare come cambiano i valori delle variabili elaborate.

Dopo l’impostazione di punti d’arresto, si può dapprima far eseguire il programma fino al primo punto d’arresto e a partire da questo punto continuare con il controllo passo a passo.

Punti di arresto

I punti d’arresto possono essere definiti in qualsiasi punto della parte istruzioni del testo sorgente.

I punti di arresto vengono trasferiti al sistema di automazione e attivati solo con la selezione del comando di menu **Test ▸ Modifica punti d’arresto**. Il programma viene quindi eseguito fino al raggiungimento del primo punto di arresto.

Il numero dei punti di arresto attivi dipende dalla CPU:

- CPU 416: sono possibili max. 4 punti di arresto attivi
- CPU 414: sono possibili max. 2 punti di arresto attivi
- CPU 314: non è possibile alcun punto di arresto attivo

Funzione a passi singoli

Dopo aver avviato la funzione di test **Punti d’arresto attivi** , si possono eseguire le seguenti funzioni:

- **Esegui istruzione successiva**

  Avanzare di un’istruzione: permette l’emissione dei valori delle variabili

- **Continua**

  Continuare fino al successivo punto di arresto attivo.

- **Esegui fino alla selezione**

  Continuare fino a un punto di testo contrassegnato nella sorgente, definito dall’utente.

Avvertenza

Si deve tener presente che il numero massimo dei punti di arresto attivi non viene superato quando si usano i comandi di menu **Esegui istruzione successiva** o **Esegui fino alla selezione**, poiché tali comandi impostano e attivano implicitamente un punto di arresto.

Come si usa la funzione ”Punti d’arresto attivi”?

Prima di avviare il test, assicurarsi che siano soddisfatti i requisiti indicati nel capitolo 6.1. A questo punto è possibile testare il programma nella CPU passo dopo passo utilizzando la funzione ”Punti d’arresto attivi”. Il procedimento viene illustrato nella descrizione dettagliata che segue e nella figura 6-1:

1. Selezionare la finestra contenente la sorgente del blocco da testare.
2. Impostare i punti di arresto posizionando il cursore sulla posizione desiderata nella sorgente programma e scegliere quindi il comando di menu **Test ▸ Definisci punto d’arresto**. I punti di arresto vengono rappresentati con un cerchio rosso sul margine sinistro della finestra
3. Avviare l’esecuzione a singoli passi selezionando il menu **Test ▸ Punti d’arresto attivi**.

   Risultato: la finestra viene suddivisa in due in senso verticale e il programma viene avviato. Inizia la ricerca del successivo punto di arresto. Non appena raggiunto, la CPU passa allo stato di funzionamento ALT mentre lo sfondo del punto di arresto rosso viene marcato in grigio.
4. Da qui in poi, per l’esecuzione in singoli passi, sono disponibili le seguenti funzioni alternative:
   – Selezionare il comando di menu **Test ▸ Esegui istruzione successiva** (4a)

     Risultato: la CPU commuta allo stato di funzionamento RUN. Dopo aver elaborato l’istruzione immediatamente successiva si interrompe nuovamente e visualizza nella finestra risultati i valori delle variabili elaborati nell’istruzione precedente.
   – Selezionare il comando di menu **Test ▸ Continua** (4b).
     Risultato: la CPU commuta allo stato di funzionamento RUN. Dopo aver raggiunto il punto d’arresto immediatamente successivo si interrompe nuovamente e visualizza nella finestra risultati il punto di arresto. Per visualizzare i valori delle variabili selezionare nuovamente il comando di menu **Test ▸ Esegui istruzione successiva**.
   – Selezionare il comando di menu **Test ▸ Esegui fino alla selezione** (4c).
     Nella posizione attuale della selezione viene implicitamente impostato ed attivato un punto di arresto. La CPU commuta allo stato di funzionamento RUN. Dopo aver raggiunto la selezione si interrompe nuovamente e visualizza nella finestra risultati questo punto di arresto. Per visualizzare i valori delle variabili selezionare nuovamente il comando di menu **Test ▸ Esegui istruzione successiva**.
5. Ritornare al punto 2 se si intende continuare il test con punti di arresto diversi. Al punto 2 si possono definire nuovi punti di arresto o cancellare punti di arresto esistenti.
6. Selezionare il comando di menu **Test ▸Punti d’arresto attivi**, per interrompere il loop del test.
7. Se non si intendono testare ulteriori istruzioni nella sorgente, concludere il test selezionando il comando di menu **Test ▸Concludi test.**

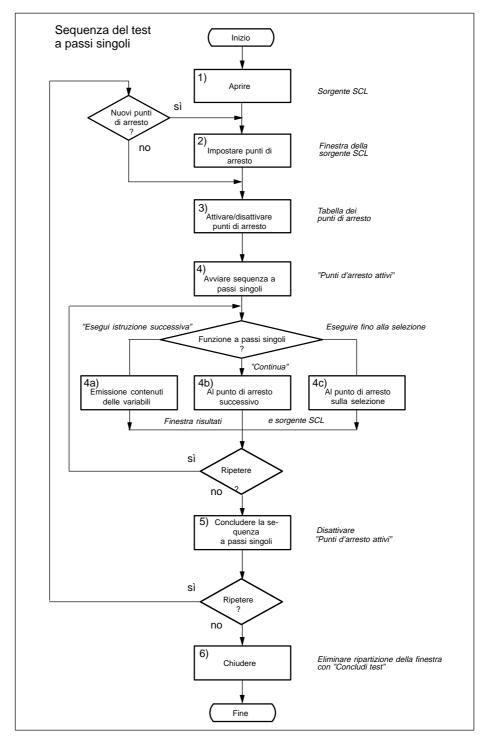



Figura 6-1 Algoritmo della sequenza di test

6.4 Funzione di test ”Controlla/comanda variabili”

Panoramica

Durante il test con la funzione ”Controlla/comanda variabili” è possibile

- fare visualizzare (controllare) i valori attuali dei dati globali del proprio programma utente
- assegnare (comandare) valori fissi alle variabili di un programma utente.

Controllo e comando delle variabili

Con il comando di menu **Sistema di destinazione** ▸ **Controlla/comanda variabili** è possibile:

- definire i punti e le condizioni di trigger
- specificare i valori per le variabili di un programma utente.

In entrambi i casi si deve creare una tabella delle variabili indicando quali variabili debbano essere elaborate. Nel caso di ”Comanda” si devono indicare anche i valori desiderati.

Il manuale utente STEP 7 **/231/** contiene una descrizione dettagliata della funzione di test.

6.5 Funzione di test ”Dati di riferimento”

Panoramica

È possibile generare e valutare i dati di riferimento per agevolare l’esecuzione del test e delle modifiche del proprio programma utente.

I dati di riferimento comprendono: struttura del programma, elenco di riferimenti incrociati, tabella di occupazione, elenco degli operandi non utilizzati ed elenco degli operandi senza simbolo.

I dati di riferimento possono essere utilizzati come

- panoramica di un intero programma utente
- base per modifiche e test
- integrazione della documentazione del programma.

Creazione dei dati di riferimento

Per creare i dati di riferimento si hanno le seguenti possibilità:

- Tramite il comando di menu **Strumenti ▸ Dati di riferimento** è possibile eventualmente creare, aggiornare e visualizzare i dati di riferimento.
- Tramite il comando di menu **Strumenti ▸ Impostazioni** è possibile stabilire che i dati di riferimento vengano creati automaticamente durante la compilazione di una sorgente. Selezionare a tal fine nella scheda ”Crea blocco” la voce ”Crea dati di riferimento”.
  La creazione automatica dei dati di riferimento allunga tuttavia la procedura di compilazione.

Il manuale utente STEP 7 **/231/** contiene una descrizione dettagliata della funzione di test.

6.6 Uso delle funzioni di test STEP 7

Editor AWL

I blocchi compilati con SCL possono essere aperti in STEP 7 e quindi testati con l’editor AWL.

Interrogazione e modifica dello stato di funzionamento della CPU

Selezionare il comando di menu **Sistema di destinazione ▸ Stato di funzionamento**, per interrogare o modificare l’attuale stato di funzionamento della CPU.

Visualizzazione delle proprietà della CPU

Il comando di menu **Sistema di destinazione ▸ Stato dell’unità** apre una finestra di dialogo in cui si può:

- determinare la causa della transizione in STOP mediante lettura del buffer di diagnostica.
- interrogare il contenuto della CPU. In particolare, lo stack delle interruzioni è un importante aiuto per la ricerca degli errori.
- avere informazioni sui dati tecnici della CPU.
- leggere l’ora e la data della CPU.
- definire il tempo di ciclo della CPU.
- avere informazioni sui blocchi contenuti nella CPU.
- avere informazioni sulla comunicazone della CPU.

Per poter eseguire queste funzioni la CPU deve trovarsi nel modo online.

Parte 3: Descrizione del linguaggio

Terminologia generale SCL 7

Panoramica

Questo capitolo contiene una descrizione dei vari mezzi linguistici messi a disposizione da SCL e delle relative modalità d'uso. Si prega di tener presente che qui vengono descritti solo i concetti base e le definizioni necessarie, che saranno oggetto di dettagliata descrizione nei capitoli seguenti.

Sommario del capitolo

7.1 Mezzi ausiliari per la descrizione del linguaggio

Descrizione del linguaggio di SCL

La base per la descrizione del linguaggio è costituita dai diagrammi sintattici. Essi offrono una chiara panoramica della struttura sintattica (ossia grammaticale) di SCL. L'appendice B del manuale contiene un riepilogo di tutti i diagrammi con gli elementi linguistici.

Che cos'è un diagramma sintattico?

Il diagramma sintattico è una rappresentazione grafica della struttura del linguaggio. La struttura è costituita da una sequenza gerarchica di regole. Ogni regola a sua volta è presente come blocco in una regola sottostante.

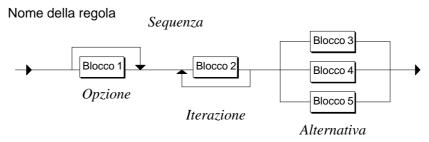



Figura 7-1 Diagramma sintattico

Il diagramma sintattico viene letto da destra verso sinistra. Si devono osservare le seguenti strutture delle regole:

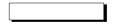
- Sequenza: sequenza di blocchi
- Opzione: ramo saltabile
- Iterazione: ripetizione di rami
- Alternativa: diramazione

Quali tipi di blocchi esistono?

Un blocco è un elemento base oppure un elemento che a sua volta si compone di vari blocchi. La figura seguente mostra i tipi di simboli corrispondenti ai vari blocchi:

Elemento base che non deve essere ulteriormente descritto.

Si tratta di caratteri stampabili e di caratteri speciali, di parole chiave e di identificatori predefiniti.

Le indicazioni su questi blocchi devono essere accettate senza alcuna modifica.

Elemento composto che viene descritto da ulteriori diagrammi sintattici.

Che cosa significa libertà di formato?

Durante l'introduzione di testi sorgente si devono osservare sia le **regole sintattiche** che le **regole lessicali**.

Entrambi i gruppi vengono descritti nelle appendici B e C. La libertà di formato significa che fra i blocchi vuoti si possono inserire caratteri di formattazione come spazi, tabulatori, cambi pagina e commenti.

Regole lessicali

Le regole lessicali, come p. es. rappresentato nella la figura 7-2 **non** consentono alcuna libertà di formato. Quando si applica una regola lessicale, le introduzioni devono essere accettate senza alcuna modifica.

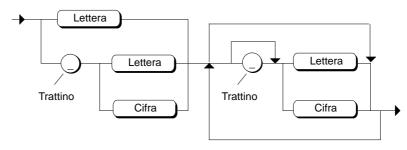



Figura 7-2 Esempio di regola **lessicale**

In base alla regola illustrata, sono esempi validi:

```
R_REGOLATORE3
_A_CAMPO
_100_3_3_10
```

Per i motivi suddetti, i seguenti esempi non sono validi:

```
1_1AB
RR__20
*#AB
```

Regole sintattiche

L'introduzione di testi sorgente conformemente alle regole sintattiche avviene con libertà di formato (vedere figura 7-3).

Figura 7-3 Esempio di regola **sintattica**

In base alla regola illustrata, sono esempi validi:

```
VARIABLE_1  := 100;     INTERRUTTORE:=FALSE;
            VARIABILE_2 := 3.2;
```

7.2 Set di caratteri SCL

Lettere, cifre

Dall’area parziale del set di caratteri ASCII, SCL utilizza:

- le lettere (maiuscole e minuscole) dalla A alla Z,
- le cifre arabe da 0 a 9,
- i blank (spazi, valore ASCII 32) e tutti i caratteri di controllo (ASCII 0-31) compreso il carattere di fine riga (ASCII 13).

Altri caratteri

In SCL questi caratteri hanno un significato chiaramente definito:

+	-	*	/	=	<	>	[	]	(	)
.	,	:	;	$	#	”	’	{	}	

Ulteriori informazioni

L’appendice A contiene un elenco dettagliato di tutti i caratteri utilizzabili nonché le relative informazioni indicanti il modo in cui SCL interpreta i singoli caratteri.

7.3 Parole riservate

Significato

Le parole riservate sono parole chiave che devono essere usate esclusivamente nel modo prescritto. Non viene fatta alcuna distinzione fra lettere maiuscole e minuscole.

Parole chiave

AND
ANY
ARRAY
BEGIN
BLOCK_DB
BLOCK_FB
BLOCK_FC
BLOCK_SDB
BLOCK_SFB
BLOCK_SFC
BOOL
BY
BYTE

CASE
CHAR
CONST
CONTINUE
COUNTER
DATA_BLOCK
DATE
DATE_AND_TIME
DINT
DIV
DO
DT
DWORD
ELSE
ELSIF
END_CASE
END_CONST
END_DATA_BLOCK
END_FOR
END_FUNCTION

END_STRUCT
END_VAR
END_WHILE
EXIT
FOR
FUNCTION
FUNCTION_BLOCK
GOTO
IF
INT
LABEL
MOD
NIL
NOT
OF
OR
ORGANIZATION_BLOCK
POINTER
REAL
REPEAT
RETURN
S5TIME
STRING
STRUCT
THEN
TIME
TIMER
TIME_OF_DAY
TO
TOD
TYPE
VAR
VAR_TEMP
UNTIL

Parole chiave, continuazione

END_FUNCTION_BLOCK
END_IF
END_LABEL
END_TYPE
END_ORGANIZATION_BLOCK
END_REPEAT
VOID
VAR_INPUT
VAR_IN_OUT
VAR_OUTPUT
WHILE
WORD
XOR

Altri nomi riservati

EN
ENO
OK
TRUE
FALSE
Nomi delle funzioni standard [1)]

[1)] *Nelle funzioni standard si distingue fra caratteri maiuscoli e minuscoli.*

7.4 Identificatori in SCL

Definizione

Un identificatore è un nome che l'utente può assegnare ad un oggetto linguistico di SCL, cioè ad una costante, una variabile, una funzione e a un blocco.

Regole

Gli identificatori possono essere composti con lettere o cifre in qualsiasi sequenza, purché il primo carattere sia una lettera o un trattino. Sono ammesse sia lettere maiuscole che minuscole. Anche in questo caso non viene fatta una distinzione tra maiuscole e minuscole (Anna e AnNa, p. es., saranno identici).

Un identificatore può essere rappresentato formalmente con il seguente diagramma sintattico.

IDENTIFICATORE

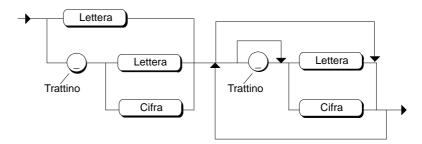



Figura 7-4 Sintassi di un identificatore

Si prega di osservare le note seguenti:

- Al momento dell'assegnazione dei nomi, si raccomanda di scegliere nomi univoci ed esplicativi, che facilitano la comprensione del programma.
- Si deve verificare se il nome è già occupato da una parola chiave (vedere tabella 7-1) o da un identificatore standard.
- La lunghezza max. di un identificatore è di 24 caratteri.
- I nomi simbolici per i blocchi (altri identificatori rispetto alla tabella 7-1) devono essere definiti nella tabella dei simboli di STEP 7 (informazioni al riguardo sono contenute in **/231/**)

Esempi

I nomi seguenti sono alcuni esempi di identificatori standard.

`x`	`y12`	`Somma`	`Temperatura`
`Nome`	`Superficie`	`Regolatore`	`Tabella`

I nomi seguenti **non** sono identificatori validi per i motivi indicati.

`4.`	Il primo carattere deve essere una lettera o un trattino.
`Array`	ARRAY è una parola chiave e non è ammessa.
`Valore S`	I blank (spazi) non sono consentiti (si deve ricordare che uno spazio è un carattere).

7.5 Identificatori standard

Definizione

In SCL è già stata definita una serie di identificatori, per i quali si usa il termine di *identificatori standard*:

- le parole chiave di blocchi e
- gli identificatori di operandi per indirizzare le aree di memoria della CPU.

Parole chiave di blocchi

Questi identificatori standard vengono utilizzati per l'indirizzamento assoluto di blocchi.

La tabella 7-1 è ordinata in base al mnemonico tedesco; inoltre, viene indicata anche il corrispondente mnemonico internazionale.

Tabella 7-1 Parole chiave di blocchi

Mnemonico SIMATIC	Mnemonico IEC	identifica
DBx	DBx	Blocco dati (Data-Block)
FBx	FBx	Blocco funzionale (Function-Block)
FCx	FCx	Funzione (Function)
OBx	OBx	Blocco organizzativo (Organization-Block)
SDBx	SDBx	Blocco dati di sistema (System- Data-Block)
SFCx	SFCx	Funzione di sistema (System-Function)
SFBx	SFBx	Blocco funzionale di sistema (System-Function-Block)
Tx	Tx	Temporizzatore (Timer)
UDTx	UDTx	Tipo di dati definito dall'utente (User defined Data Type)
Zx	Cx	Contatori (Counter)
x = Cifra compresa fra 0 e 65533 DB0 = Riservato!		

IDENTIFICATORE STANDARD

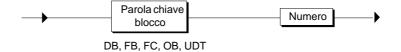



Figura 7-5 Sintassi di un identificatore standard

Sono identificazioni valide:

```
FB10
DB100
T141
```

Identificatore di operandi

Le aree di memoria di una CPU possono essere indirizzate da qualsiasi posizione del programma con il proprio identificatore di operando.

La tabella seguente è ordinata in base al mnemonico SIMATIC; inoltre, viene indicata anche il corrispondente mnemonico internazionale.

Mnemonico SIMATIC	Mnemonico IEC	indirizza	Tipo di dati
Ax,y	Qx,y	Uscita (tramite immagine di processo)	Bit
ABx	QBx	Uscita (tramite immagine di processo)	Byte
ADx	QDx	Uscita (tramite immagine di processo)	Doppia parola
AWx	QWx	Uscita (tramite immagine di processo)	Parola
AXx.y	QXx.y	Uscita (tramite immagine di processo)	Bit
Dx.y [1]	Dx.y [1]	Blocco dati	Bit
DBx [1]	DBx [1]	Blocco dati	Byte
DDx [1]	DDx [1]	Blocco dati	Doppia parola
DWx [1]	DWx [1]	Blocco dati	Parola
DXx	DXx	Blocco dati	Bit
Ex.y	Ix.y	Ingresso (tramite immagine di processo)	Bit
EBx	IBx	Ingresso (tramite immagine di processo)	Byte
EDx	IDx	Ingresso (tramite immagine di processo)	Doppia parola
EWx	IWx	Ingresso (tramite immagine di processo)	Parola
EXx.y	IXx.y	Ingresso (tramite immagine di processo)	Bit
Mx.y	Mx.y	Merker	Bit
MBx	MBx	Merker	Byte
MDx	MDx	Merker	Doppia parola
MWx	MWx	Merker	Parola
MXx.y	MXx.y	Merker	Bit
PABx	PQBx	Uscita (periferia diretta)	Byte
PADx	PQDx	Uscita (periferia diretta)	Doppia parola
PAWx	PQWx	Uscita (periferia diretta)	Parola
PEBx	PIBx	Ingresso (periferia diretta)	Byte
PEDx	PIDx	Ingresso (periferia diretta)	Doppia parola
PEWx	PIWx	Ingresso (periferia diretta)	Parola
PEWx	PIWx	Ingresso (periferia diretta)	Parola
PEWx	PIWx	Ingresso (periferia diretta)	Parola
x = Numero compreso fra 0 e 65535 (indirizzo assoluto) y = Numero compreso fra 0 e 7 (numero di bit)			

Sono esempi validi:

```
E1.0        MW10        PAW5        DB20.DW3
```

[1)] *Questi identificatori di operando valgono solo insieme all'indicazione del blocco dati.*

7.6 Numeri

Panoramica

In SCL i numeri possono essere scritti in vari modi. Un numero può avere, a scelta, un segno, un punto decimale o un esponente. Le seguenti asserzioni valgono per tutti i numeri:

1. Un numero non può contenere né virgole né spazi.
2. Per una separazione ottica, è consentito l'uso di un trattino (_).
3. Un numero può essere preceduto a scelta da un segno positivo (+) o negativo (-). Se nessun segno precede il numero, quest'ultimo viene considerato positivo.
4. I numeri non devono essere superare determinati valori massimi e minimi.

Numero intero

Un numero intero non contiene né un punto decimale né un esponente. Perciò, un numero intero è una semplice sequenza di cifre che può iniziare con un segno matematico. In SCL sono stati integrati 2 tipi di numeri interi, i quali hanno diversi campi di valori, `INT` e `DINT` (vedere capitolo 9)

Seguono alcuni numeri interi validi :

```
0          1          +1         -1
743        -5280      600_00     -32_211
```

Per i motivi suddetti, i seguenti numeri interi sono **falsi**:

`123,456`	Non sono ammesse virgole.
`36.`	Un numero intero non può contenere alcun punto decimale.
`10 20 30`	Non sono consentiti spazi.

Numeri interi come numeri binari, ottali o esadecimali

In SCL, i numeri interi possono essere rappresentati con un altro sistema numerico. Ciò avviene facendo precedere al numero una parola chiave per il sistema numerico. Per esempio 2# indica il sistema binario, 8# il sistema ottale e 16# il sistema esadecimale.

Seguono alcuni numeri interi validi per il numero decimale 15:

```
2#1111     8#17       16#F
```

Numeri in virgola mobile

Un numero in virgola mobile deve contenere o un punto decimale o un esponente (o entrambi). Un punto decimale deve essere situato fra due cifre. Perciò, un numero in virgola mobile non può né iniziare né terminare con un punto decimale.

Seguono alcuni numeri in virgola mobile validi:

```
0.0        1.0        - 0.2      827.602
50000.0    -0.000743  12.3       -315.0066
```

I seguenti numeri reali sono **falsi**:

`1.`	Il punto decimale deve essere situato fra due cifre.
`1,000.0`	Non sono ammesse virgole.
`.3333`	Il punto decimale deve essere situato fra due cifre.

Un esponente può essere presente per definire la posizione del punto decimale. Se non è presente alcun punto decimale, si presuppone che esso sia situato a destra della cifra. L'esponente deve essere un numero intero positivo o negativo. La base 10 viene sostituita con la lettera E.

In SCL il valore 3×10^{10} può essere rappresentato con i seguenti numeri in virgola mobile:

`3.0E+10`	`3.0E10`	`3e+10`	`3E10`
`0.3E+11`	`0.3e11`	`30.0E+9`	`30e9`

I seguenti numeri in virgola mobile sono **falsi**:

`3.E+10`	Il punto decimale deve essere situato fra due cifre.
`8e2.3`	L'esponente deve essere un numero intero.
`.333e-3`	Il punto decimale deve essere situato fra due cifre.
`30 E10`	Non sono ammessi spazi.

Stringhe di caratteri

Una stringa di caratteri è una sequenza di caratteri (cioè lettere, cifre e caratteri speciali) racchiusi fra virgolette. Si possono utilizzare sia lettere maiuscole che minuscole.

Seguono alcune stringhe di caratteri valide :

`'ROT'`	`'76181 Karlsruhe'`	`'270-32-3456'`
`'DM19.95'`	`'La risposta esatta è:'`	

Con il simbolo $ si possono introdurre speciali caratteri di formattazione, le virgolette (') o un carattere $.

Testo sorgente	dopo la compilazione
`'SIGNAL$'ROT$''`	`SIGNAL'ROT'`
`'50.0$$'`	`50.0$`
`'VALORE$P'`	`VALORE` *Rimpagina*
`'REG-$L'`	`REG-`*A capo automatico*
`'REGEL$R`	`REGLER`*Ritorno del carrello*
`'SCHRITT$T'`	`SCHRITT` *Tabulatore*

Per i caratteri non stampabili si deve introdurre la rappresentazione sostitutiva in codice esadecimale con $hh, in cui *hh* indica il valore in forma esadecimale del carattere ASCII.

Per interrompere una stringa di caratteri (per commenti che non devono essere stampati o visualizzati) in SCL sono disponibili i caratteri $> e $< che consentono l'interruzione di una stringa.

7.7 Tipi di dati

Panoramica

In ogni dichiarazione di variabili si deve indicare il tipo di variabile. Il tipo definisce il campo di valori delle variabili e determina le operazioni che possono essere eseguite con le variabili.

Un certo tipo di dati determina :

- il tipo ed il significato di un elemento di dati,
- il campo di valori consentito dell'elemento di dati,
- la quantità consentita delle operazioni che possono essere eseguite con un operando di un tipo di dati
- la modalità di scrittura dei dati di questo tipo.

Tipi di dati

Si distinguono i seguenti tipi di dati.

Tabella 7-2 Tipi di dati semplici

Tipi di dati	Significato
Semplici	Fa parte della dotazione standard di SCL.
Composti	Si possono generare mediante associazione di tipi di dati semplici.
Definiti dall'utente	Vengono definiti appositamente per la propria applicazione con un nome scelto a piacere.
Tipi di parametri	Possono essere impiegati solo per la definizione di parametri.

Tipi di dati semplici

I tipi di dati semplici definiscono la struttura di dati che non possono essere frazionati in unità più piccole. Essi sono conformi alle definizione della norma DIN EN 1131-3.

In SCL sono stati predefini 12 tipi di dati semplici:

BOOL	CHAR	INT	TIME
BYTE		DINT	DATE
WORD		REAL	TIME_OF_DAY
DWORD			S5TIME

Tipi di dati composti

I tipi di dati composti definiscono strutture di dati che che si compongono di altri tipi di dati. SCL consente i seguenti tipi di dati composti:

DATE_AND_TIME

STRING

ARRAY

STRUCT

Tipi di dati definiti dall’utente

Si tratta di tipi di dati globali (UDT), che l’utente può creare in SCL per le proprie applicazioni. Questo tipo di dati può essere utilizzato con la propria identificazione UDT (x indica il numero di serie) oppure sotto un nome simbolico ad esso associato e definito nella parte convenzioni di un blocco o di un blocco dati.

Tipi di parametri

Oltre ai tipi di dati semplici, composti e definiti dall’utente, si possono utilizzare tipi di parametri per la definizione di parametri. A tal fine SCL mette a disposizione i seguenti parametri:

TIMER	BLOCK_FB	POINTER	ANY
COUNTER	BLOCK_FC		
	BLOCK_DB		
	BLOCK_SDB		

7.8 Variabili

Dichiarazione di variabili

Un identificatore, il cui valore può essere modificato durante l'esecuzione del programma, viene denominato *Variabile*. Ogni variabile deve essere dichiarata (ossia definita) singolarmente prima che possa essere utilizzata in un blocco di codice o in un blocco dati. Nella dichiarazione delle variabili si definisce che un identificatore è una variabile (e non una costante, ecc.) e si specifica il tipo di variabile mediante associazione ad un tipo di dati.

A seconda della validità delle variabili si distinguono:

- dati locali
- dati utente globali
- variabili consentite e predefinite (aree di memoria nella CPU)

Dati locali

I dati locali sono dati utilizzati nell'ambito di un blocco di codice (FC, FB, OB) e sono validi solo per questo blocco di codice. Si tratta in particolare dei dati seguenti:

Tabella 7-3 Dati locali di un blocco

Variabli	Significato
Variabili statiche	Una variabile statica è un variable locale, il cui valore rimane costante durante tutte le esecuzioni del blocco (memoria del blocco). Tale variabile serve per momorizzare i valori di un **blocco funzionale**.
Variabili temporanee	Le variabili temporanee fanno parte di un blocco di codice e non occupano **alcun** campo di memoria statico. Il loro valore rimane costante solo durante l'esecuzione di un blocco. Alle variabili temporanee **non** si può accedere al di fuori del blocco in cui è stata dichiarata la variabile.
Parametri del blocco	Parametri del blocco sono parametri formali di un blocco funzionale o di una funzione. Si tratta di variabili locali che servono per trasferire i parametri attuali indicati al momento del richiamo.

Dati utente globali

I dati utente globali sono dati risp. aree di dati che possono essere utilizzati in qualsiasi parte del programma. A tal fine si devono creare blocchi dati (DB).

Quando si crea un DB, la sua struttura viene definita in un'apposita convenzione. Al posto di una convenzione di struttura si può anche utilizzare un tipo di dati definito dall'utente (UDT). La sequenza in cui l'utente introduce i componenti della struttura determina la sequenza dei dati nel DB.

Aree di memoria di una CPU

Alle aree di memoria di una CPU si può accedere tramite gli identificatori di operandi (vedere capitolo 7.5) direttamente da un punto qualsiasi del programma, senza dover prima definire queste variabili.

Si ha inoltre la possibilità di indirizzare queste aree di memoria anche simbolicamente. In tal caso l'assegnazione dei simboli avviene in modo globale nella tabella dei simboli in STEP 7. Per ulteriori informazioni al riguardo si rimanda a **/231/**.

7.9 Espressioni

Panoramica

Un'espressione contiene un valore che viene calcolato in fase di compilazione oppure durante l'esecuzione del programma. Esso si compone di uno o più operandi che vengono associati fra loro mediante operatori. La sequenza di analisi degli operatori viene indicata dalla loro priorità e può inoltre essere determinata mediante parentesi.

- Espressioni aritmetiche
- Espressioni logiche
- Espressioni di confronto

Espressione aritmetica

Una tipica espressione è p. es.:

```
(b*b-4*a*c)/(2*a)
```

Gli identificatori a e b nonché le cifre 4 e 2 sono gli operandi; i simboli *, - e / sono i corrispondenti operatori (moltiplicazione, sottrazione e divisione). L'intera espressione rappresenta una cifra.

Espressioni di confronto

Un'espressione di confronto è un'espressione logica che può essere vera o falsa. Segue un esempio di espressione di confronto:

```
Valore di riferimento < 100.0
```

In questa espressione, `VALORE DI RIFERIMENTO` è una variabile, 100.0 un numero in virgola mobile e il simbolo < un operatore di confronto. L'espressione ha il valore **vero** se il valore di riferimento rappresenta un valore minore di 100.0, altrimenti l'espressione ha il valore **falso**.

Espressione logica

Un esempio tipico è:

```
a AND NOT b
```

Gli identificatori a e b sono gli operandi; le parole chiave AND e NOT sono gli operatori logici. L'intera espressione rappresenta una stringa di bit.

7.10 Istruzioni

Panoramica

Un'istruzione SCL è un'azione eseguibile nella parte istruzioni di un blocco di codice. In SCL esistono tre istruzioni fondamentali:

1. Assegnazioni di valori (assegnazione di un'espressione ad una variabile)
2. Istruzioni di controllo (ripetizione o diramazione di istruzioni)
3. Elaborazione di sottoprogrammi (richiamo o diramazione di blocchi di codice)

Assegnazioni di valori

Una tipica assegnazione di valori p. es. è la seguente:

```
VALORE_DI_RIFERIMENTO := 0.99*VALORE_DI
RIFERIMENTO_VECCHIO
```

In questo esempio si presuppone che `VALORE_DI_RIFERIMENTO` e `VALORE_DI_RIFERIMENTO_VECCHIO` siano variabili in virgola mobile. Il valore di assegnazione moltiplica il valore `VALORE_DI_RIFERIMENTO_VECCHIO` per 0.99 e assegna il risultato alla variabile `VALORE_DI_RIFERIMENTO`. Si deve tener presente che il simbolo per l'assegnazione è :=.

Istruzioni di controllo

Una tipica istruzione di controllo è la seguente:

```
FOR Contatore :=1 TO 20 DO
            LISTA[Contatore]        := VALORE+Contatore;
END_FOR;
```

In questo esempio l'assegnazione viene eseguita 20 volte. Ogni volta, nel campo `LISTA` il nuovo valore calcolato viene trascritto in un posto della lista immediatamente superiore.

Elaborazione di sottoprogrammi

Con l'indicazione di un'identificazione di blocco per una funzione (FC) o per un blocco funzionale (FB), viene richiamato il blocco di codice che è stato dichiarato con questa identificazione.[1)] Se la dichiarazione del blocco di codice contiene parametri formali, durante il richiamo dei parametri formali gli operatori attuali possono essere assegnati ai parametri formali.

Tutti i parametri elencati nel blocco dichiarazioni

```
VAR_INPUT, VAR_OUTPUT e VAR_IN_OUT
```

di un blocco di codice vengono definiti parametri formali - i corrispondenti parametri nei richiami nell'ambito della parte istruzioni vengono invece denominati parametri attuali.

Il trasferimento dei parametri attuali ai parametri formali fa parte integrante del richiamo.

Una tipica elaborazione di sottoprogramma è p. es. la seguente:

```
FC31(X:=5, Q1:=Somma di controllo);
```

[1)] *Se sono stati dichiarati dei parametri formali in una FC, occorre altretanto assegnare dei parametri attuali. Per gli FB non si applica la stessa regola.*

7.11 Blocchi SCL

Panoramica

In un file sorgente SCL si possono programmare da 1 fino a n blocchi come testo sorgente.

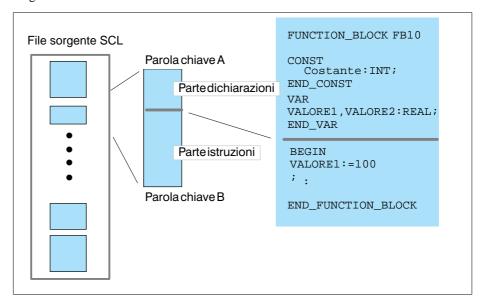
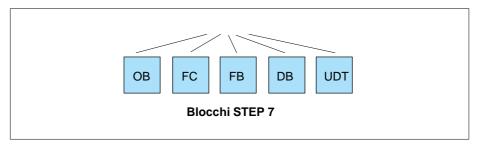



Figura 7-6 Struttura di un file sorgente SCL

Tipi di blocchi

Per la loro funzione, struttura o tipo d'impiego, i blocchi STEP 7 sono parti ben definite di un programma utente. Con SCL si possono programmare i seguenti blocchi:

Blocchi predefiniti

L'utente non deve programmare ogni singola funzione. Si possono impiegare anche blocchi predefiniti. Essi sono presenti nel sistema operativo dell'unità centrale o nelle biblioteche *(S7lib)* del pacchetto base STEP 7 e possono essere utilizzati p. es. per programmare funzioni di comunicazione.

Struttura di un blocco SCL

Ogni blocco si compone dei seguenti componenti:

- Inizio/fine intestazione del blocco (parola chiave in base al tipo di blocco)
- Parte dichiarazioni
- Parte istruzioni (nei blocchi dati parte assegnazioni)

Parte dichiarazioni

Nella parte dichiarazioni devono essere descritte tutte le definizioni necessarie come base per la parte istruzioni: p. es. definizione di costanti e dichiarazione di variabili e parametri.

Parte istruzioni

La parte istruzioni inizia con la parola chiave `BEGIN` e termina con un identificatore standard per la fine del blocco `END_xxx` (vedere capitolo 8.2).

Ogni istruzione termina con un punto e virgola (” ; ”). Come opzione, prima di ogni istruzione vi può essere un’etichetta di salto (Label). Per la sintassi della parte istruzioni e delle istruzioni singole si rimanda al capitolo 13.

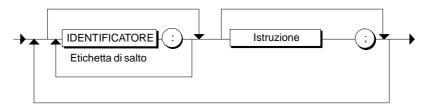



Figura 7-7 Sintassi di un’istruzione

Segue un esempio per la parte istruzioni di un FB:

```
:                  //Fine parte dichiarazioni
:
BEGIN              //Inizio della parte istruzioni
            X := X+1;
LABEL1      Y := Y+10;
            Z := X*Y;
            :
            GOTO LABEL1
LABELn;     FC10 := Z;//Fine della parte istruzioni
END_FUNCTION_BLOCK
```

Nella parte istruzioni di un blocco dati, i dati DB possono essere impostati con determinati valori mediante assegnazioni di valori. Perciò, nei capitoli seguenti la parte istruzioni di un DB viene denominata **parte assegnazioni**.

Programma S7

Dopo la compilazione, i blocchi generati vengono depositati nel contenitore ”Blocchi” del relativo programma S7. Da qui essi devono essere trasferiti nella CPU. Per informazioni al riguardo si rimanda a **/231/** .

7.12 Commenti

Panoramica

I commenti servono per la documentazione e la migliore comprensione di un blocco SCL. Dopo la fase di compilazione, essi non hanno alcuna rilevanza ai fini dell'esecuzione del programma. Esistono due tipi di commenti:

- il commento a una riga
- il commento a più righe

Commento a una riga

Il commento a una riga viene preceduto dai caratteri // e va fino alla fine di una riga. La sua lunghezza è limitata a max. 253 caratteri compreso il carattere introduttivo //. Esso può essere rappresentato in modo formale con il seguente diagramma sintattico:

Commento a una riga

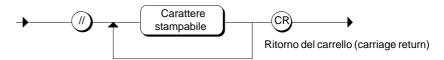



Figura 7-8 Sintassi di un commento a una riga

Per i caratteri stampabili si rimanda alla tabella A-2 nell'appendice. Nell'ambito di un commento a una riga, la coppia di caratteri '(*' e '*)' non ha alcuna rilevanza.

Commento a più righe

Commento che può estendersi su diverse righe e viene introdotto come blocco con '(*' e terminato con '*)'.

Commento a più righe

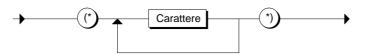



Figura 7-9 Sintassi di un commento a più righe

Per i caratteri consentiti si rimanda alla tabella A-2 in appendice.

Ciò che si deve osservare:

Quando vengono introdotti dei commenti si devono osservare i punti seguenti:

- I commenti a più righe nei **blocchi dati** devono essere introdotti come commenti alle righe, ovvero tali commenti vengono iniziati con ’//’.
- Nello standard l’annidamento di commenti è ammesso. Questa impostazione del compilatore può tuttavia essere modificata con l’opzione ”Autorizza commenti annidati”. In tal caso selezionare il comando di menu **Strumenti ▸ Impostazioni**, quindi selezionare l’opzione nella scheda ”Compilatore” della finestra di dialogo successiva.
- Un commento non deve interrompere né un nome simbolico né una costante. Però è consentita l’interruzione di stringhe.

Questi commenti sono **falsi**:

```
(*// FUNCTION_BLOCK // Adattamento *) BLOCK FB 10
```

Esempio di inserimento di commenti

Nell’esempio sono contenuti due blocchi al commento e un commento a una riga.

```
FUNCTION_BLOCK FB15
(* Qui vi è il commento a più righe,
che può estendersi su diverse righe*)
VAR
   INTERRUTTORE: INT; // commento a una riga
END_VAR;
BEGIN
  (* Assegnazione di un valore alla variabile *)
  INTERRUTTORE:= 3;
END_FUNCTION_BLOCK
```

Figura 7-10 Esempio di commenti

Avvertenza

I commenti a una riga che si trovano nella stessa riga direttamente dietro alla dichiarazione delle variabili di un blocco vengono importati al momento della decompilazione in AWL.

Questi commenti si trovano in AWL nell’area delle interfacce, ovvero nella parte superiore della finestra (vedere anche /**231**/).

Nell’esempio riportato nella figura 7-10 viene perciò confermato il primo commento a una riga.

Struttura di sorgenti SCL 8

Panoramica

Una sorgente SCL si compone, normalmente, di testo progressivo. In una simile sorgente si possono programmare diversi blocchi. Questi ultimi possono essere OB, FB, FC, DB, o UDT.

In questo capitolo viene descritta la struttura esterna dei blocchi. I seguenti capitoli contengono una descrizione delle strutture interne.

Sommario del capitolo

8.1 Struttura

Panoramica

Una sorgente SCL si compone del testo sorgente da 1 fino a n blocchi (si tratta dei blocchi FB, FC, OB, DB e UDT).

Per poter compilare la propria sorgente SCL in singoli blocchi, l'utente deve osservare determinate strutture e norme sintattiche concernenti questi blocchi.

Unità di programma SCL

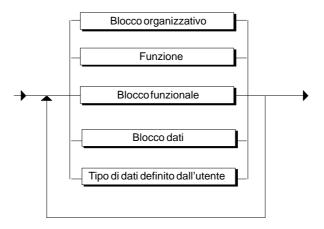



Figura 8-1 Sintassi di una sorgente

Sequenza dei blocchi

Per quanto riguarda la sequenza dei blocchi, durante la creazione della sorgente si deve osservare quanto segue:

I blocchi richiamati devono essere situati prima dei blocchi richiamanti.

Ciò significa:

- I tipi di dati definiti dall'utente (UDT) sono situati prima dei blocchi in cui essi vengono utilizzati.
- I blocchi dati ai quali è stato assegnato un tipo di dati definito dall'utente (UDT) sono situati dopo l'UDT.
- I blocchi dati a cui è possibile accedere da tutti i blocchi di codice sono situati prima di tutti i blocchi dati che accedono ad essi.
- I blocchi dati con blocco funzionale assegnato sono situati dietro il blocco funzionale.
- Il blocco organizzativo OB1 il quale richiama altri blocchi, è situato alla fine dei blocchi. A loro volta, i blocchi che vengono richiamati da blocchi richiamati dall'OB1 devono essere situati prima di questi ultimi blocchi.

I blocchi che vengono richiamati nel file sorgente ma che non vengono programmati nello stesso file sorgente devono essere già presenti al momento della compilazione del file nel relativo programma utente.

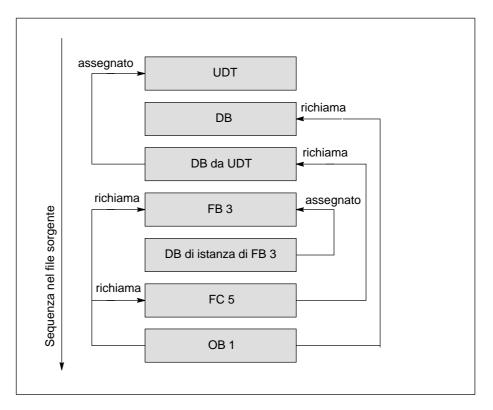



Figura 8-2 Esempio di struttura di blocchi di un file sorgente

Struttura generica dei blocchi

Il codice sorgente di un blocco consiste essenzialmente delle parti seguenti:

- inizio blocco con indicazione del blocco (assoluta o simbolica)
- attributi di blocco (opzionale)
- parte convenzioni (diversa a seconda del tipo di blocco)
- parte istruzioni nei blocchi di codice o
  assegnazione di valori attuali in blocchi dati (opzionale)
- fine blocco

8.2 Inizio e fine di un blocco

Panoramica

Il testo sorgente per un singolo blocco viene introdotto, in funzione del tipo di blocco, con un'identificazione standard per l'inizio del blocco alla quale precede un'identificatore del blocco e viene terminato con un'identificazione standard per la fine del blocco (vedere tabella 8-1).

Tabella 8-1 Identificazione standard per l'inizio e la fine del blocco

Sintassi

Sintassi	Tipo di blocco	Nome del blocco

`ORGANIZATION_BLOCK` *`Ob_name`*
`:`
`END_ORGANIZATION_BLOCK`
	OB	Blocco organizzativo

`FUNCTION` *`fc_name:tipo di funzione`*
`:`
`END_FUNCTION`
	FC	Funzione

`FUNCTION_BLOCK` *`ob_name`*
`:`
`END_FUNCTION_BLOCK`
	FB	Blocco funzionale

`DATA_BLOCK` *`db_name`*
`:`
`END_DATA_BLOCK`
	DB	Blocco dati

`TYPE` *`name udt_name`*
`:`
`END_TYPE`
	UDT	Tipo di dati definito dall'utente

Nome del blocco

Nella tabella 8-1 *xx_nome* indica il nome del blocco conformemente alla sintassi seguente:

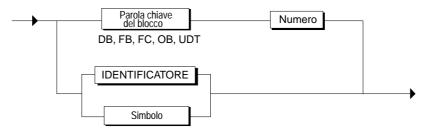



Figura 8-3 Sintassi del nome del blocco

Per ulteriori informazioni al riguardo si rimanda al capitolo 7.5. Si deve inoltre osservare che un'identificazione o un simbolo devono essere definiti nella tabella dei simboli di STEP 7 (vedere /**231**/).

Esempio

```
FUNCTION_BLOCK FB10
FUNCTION_BLOCK Bloccoregolatore
FUNCTION_BLOCK ”Regolatore.B1&U2”
```

8.3 Attributi di blocco

Definizione

Gli attributi per i blocchi possono essere:

- attributi dei blocchi
- attributi di sistema per i blocchi.

Attributi dei blocchi

Titolo, versione, protezione blocco, autore, nome e famiglia di un blocco possono essere introdotti con l'ausilio di parole chiave.

Sintassi

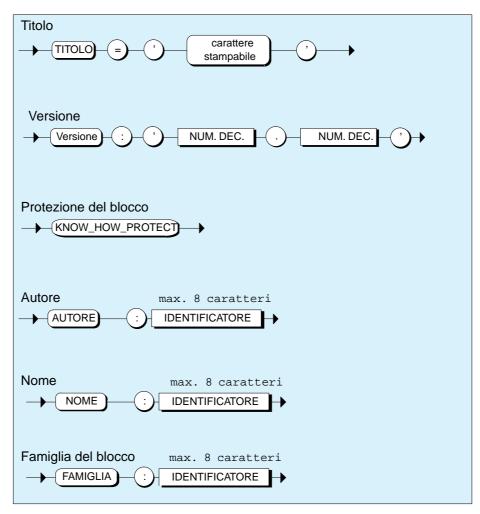



Figura 8-4   Sintassi: Attributi di blocco

Attributi di sistema per blocchi

Ai blocchi possono essere inoltre assegnati attributi di sistema, ad es. per la progettazione del controllo di processo.

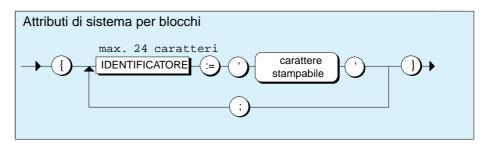



Figura 8-5 *Sintassi: Attributi di sistema per blocchi*

La tabella 8-2 mostra quali attributi di sistema per blocchi possono essere assegnati in SCL.

Tabella 8-2 Attributi di sistema per blocchi

Attributo	Valore	Assegnare questo attributo quando	Tipo di blocco ammesso
S7_m_c	true, false	il blocco deve essere comandato o controllato da un'apparecchiatura di servizio e supervisione.	FB
S7_tasklist	taskname1, taskname2, etc.	il blocco deve essere richiamato oltre che in blocchi organizzativi ciclici anche in altri OB (ad es. OB di errore o OB di avviamento).	FB, FC
S7_blockview	big, small	il blocco deve essere rappresentato in formato piccolo o grande su un'apparecchiatura di servizio e supervisione.	FB, FC

Assegnazione di attributi

Gli attributi di blocco vengono assegnati **dopo** il nome del blocco e **prima** della parte dichiarazioni. Un identificatore deve avere una lunghezza max. di 8 caratteri.

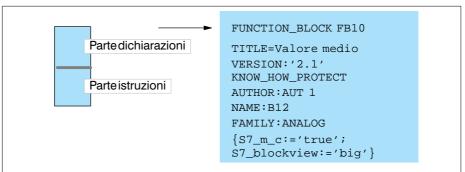



Figura 8-6 Assegnazione di attributi

8.4 Parte dichiarazioni

Panoramica

La parte dichiarazioni serve per definire le convenzioni per variabili, parametri, costanti ed etichette di salto (label).

- Le variabili locali, parametri, costanti ed etichette di salto, che devono avere validità solo all'interno di un blocco, vengono definite nella parte dichiarazioni del blocco di codice.
- I dati globali che devono poter essere indirizzati da qualsiasi blocco di codice, vengono definiti nella parte convenzioni del DB.
- Nella parte dichiarazioni di un UDT viene definito un tipo di dati definito dall'utente.

Struttura

Una parte dichiarazioni è suddivisa in vari blocchi di dichiarazione, i quali sono contrassegnati da una propria coppia di parole chiavi. Ogni blocco contiene una lista di dichiarazioni per dati simili quali costanti, etichette di salto, dati statici, dati temporanei. Un tipo di blocco deve essere presente solo una volta e conformemente alla tabella 8-3 non è consento in tutti i tipi di blocchi. La sequenza dei blocchi è irrilevante.

Tabella 8-3 Blocchi di dichiarazione consentiti

Blocchi di dichiarazione

Dati	Sintassi	FB	FC	OB	DB	UDT
Costanti	CONST *Elenco dichiarazoni* END_CONST	X	X	X		
Etichette di salto	LABEL *Elenco dichiarazoni* END_LABEL	X	X	X		
Variabili temporanee	VAR_TEMP *Elenco dichiarazoni* END_VAR	X	X	X		
Variabili statiche	VAR *Elenco dichiarazoni* END_VAR	X	X[2]		X [1]	X [1]
Parametri d'ingresso	VAR_INPUT *Elenco dichiarazoni* END_VAR	X	X			
Parametri di uscita	VAR_OUTPUT *Elenco dichiarazioni* END_VAR	X	X			
Parametri di transisto	VAR_IN_OUT *Elenco dichiarazioni* END_VAR	X	X			
Elenco dichiarazioni:	la lista degli identificatori del tipo che deve essere dichiarato.					

1 *Nei DB e UDT le parole chiave VAR e END_VAR vengono sostituite da STRUCT e END_STRUCT.*

2 *La dichiarazione di variabili all'interno della coppia di parole chiave VAR e END_VAR è ammessa nelle funzioni ma con la compilazione le dichiarazioni vengono spostate nell'area temporanea.*

Attributi di sistema per parametri

Ai parametri di ingresso, uscita e transito è possibile inoltre assegnare attributi di sistema, ad es. per la progettazione dei messaggi o dei collegamenti.

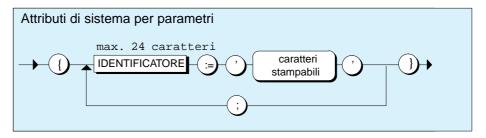



Figura 8-7 *Sintassi: Attributi di sistema per parametri*

La tabella 8-4 mostra quali attributi di sistema possono essere assegnati ai parametri:

Tabella 8-4 Attributi di sistema per parametri

Attributo	Valore	Assegnare questo attributo quando	Tipo di dichiarazione ammissibile
S7_server	connection, alarm_archiv	è rilevante il parametro per la progettazione dei collegamenti o dei messaggi. Questo parametro contiene il numero di collegamento o il numero di messaggio.	IN
S7_a_type	alarm, alarm_8, alarm_8p, alarm_s, notify, ar_send	viene definito il tipo di blocco di segnalazione che viene richiamato nella parte istruzioni (requisito fondamentale: anche l'attributo S7_server:=alarm_archiv è assegnato).	IN, solo in FB
S7_co	pbkl, pbk, ptpl, obkl, fdl, iso, pbks, obkv	viene definito il tipo di collegamento che deve essere progettato (requisito fondamentale: anche l'attributo S7_server:=connection è assegnato).	IN
S7_m_c	true, false	il parametro deve essere comandato o controllato da un'apparecchiatura di servizio e supervisione.	IN/OUT/ IN_OUT, solo in FB
S7_shortcut	2 caratteri a piacere, ad es. W, Y	al parametro devono essere assegnate sigle per la valutazione dei valori analogici.	IN/OUT/ IN_OUT, solo in FB
S7_unit	Unità, ad es. litri	al parametro devono essere assegnate unità per la valutazione dei valori analogici.	IN/OUT/ IN_OUT, solo in FB
S7_string_0	16 caratteri a piacere, ad es. APERTO	al parametro deve essere assegnato il testo per la valutazione di valori binari	IN/OUT/ IN_OUT, solo in FB, FC
S7_string_1	16 caratteri a piacere, ad es. CHIUSO	al parametro deve essere assegnato il testo per la valutazione di valori binari	IN/OUT/ IN_OUT, solo in FB, FC
S7_visible	true, false	deve o non deve essere visualizzato il parametro in CFC	IN/OUT/IN_OUT, solo in FB, FC
S7_link	true, false	il parametro deve o non deve essere interconnettibile in CFC	IN/OUT/IN_OUT, solo in FB, FC
S7_dynamic	true, false	il parametro deve o non deve essere dinamicizzabile in CFC durante l'esecuzione del test	IN/OUT/IN_OUT, solo in FB, FC
S7_param	true, false	il parametro deve o non deve essere parametrizzabile in CFC	IN/IN_OUT, solo in FB, FC

Assegnazione di attributi

Gli attributi di sistema per parametri vengono assegnati nei blocchi convenzioni parametri di ingresso, parametri di uscita o parametri di transito.

Esempio:

```
VAR_INPUT
   in1  {S7_server:='alarm_archiv';
        S7_a_type:='ar_send'}:DWORD;
END_VAR
```

8.5 Parte istruzioni

Panoramica

La parte istruzioni contiene istruzioni[1])

- che devono essere eseguite dopo il richiamo di un blocco di codice. Queste istruzioni servono per definire dati o indirizzi,
- per l'impostazione di singoli valori nei blocchi dati.

Sintassi

La figura 8-8 illustra la sintassi della parte istruzioni. Si compone della ripetizione di istruzioni singole e in cui prima di ogni istruzione vi può essere, come opzione, un'etichetta di salto (vedere capitolo 11.6) che è la destinazione di un'istruzione di salto.

Parte istruzioni

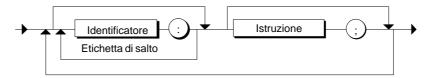



Figura 8-8 Sintassi della parte istruzioni

Sono istruzioni valide p. es.:

```
BEGIN
            VALORE INIZIALE         :=0;
            VALORE FINALE           :=200;
:
SALVARE:    RISULTATO   :=VALORE RIFERIMENTO;
:
```

Cosa occorre tener presente?

Nella formulazione delle istruzioni osservare che

- la parte istruzioni inizi con la parola chiave `BEGIN`
- la parte istruzione termini con la parola chiave per la fine del blocco
- ogni istruzione termini con un punto e virgola.
- tutti gli identificatori utilizzati nella parte istruzioni siano stati definiti in precedenza.

1) *In questo manuale si usa il termine "Istruzione" per tutti i costrutti che definiscono una funzione eseguibile.*

8.6 Istruzioni

Panoramica

Le singole istruzioni si compongono di :

- **Assegnazioni di valori** che servono per assegnare ad una variabile il risultato di un’espressione o il valore di un’altra variabile.
- **Istruzioni di controllo** che servono per ripetere istruzioni o gruppi di istruzioni oppure per realizzare diramazioni in un programma.
- **Elaborazione di sottoprogrammi** che serve per richiamare funzioni e blocchi funzionali.

Istruzione

Figura 8-9 Sintassi di una istruzione

Gli elementi necessari per formulare queste istruzioni sono espressioni, operatori e operandi. Essi vengono trattati nei capitoli seguenti.

Esempi

Gli esempi seguenti hanno lo scopo di illustrare le diverse varianti delle istruzioni:

```
// Esempio di assegnazione di valori
      VALOREMISURA:= 0 ;

// Esempio di elaborazione di sottoprogramma
      FB1.DB11(TRASFERIMENTO:= 10) ;

// Esempio di istruzione di controllo
      WHILE CONTATORE < 10 DO... ;
      :
      END_WHILE
```

Esempio 8-1 Istruzioni

8.7 Struttura di un blocco funzionale (FB)

Panoramica

Un blocco funzionale (FB) è un blocco di codice contenente una parte di un programma e che dispone di un'area di memoria assegnata. Ogni volta che viene richiamato un FB, ad esso deve essere assegnato un DB di istanza (vedere capitolo 10). L'utente determina la struttura di questo DB di istanza mediante definizione della parte dichiarazioni del DB.

Blocco funzionale

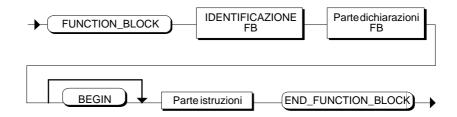



Figura 8-10 Struttura di un FB

Identificazione FB

Dopo la parola chiave

```
FUNCTION_BLOCK
```

introdurre come nome dell'FB la parola chiave FB seguita dal numero del blocco oppure il nome simbolico dell'FB.

Esempi:

```
FUNCTION_BLOCK FB10

FUNCTION_BLOCK MOTOR_1
```

Parte dichiarazioni FB

La parte dichiarazioni FB serve per la dichiarazione dei dati specifici del blocco. Per i blocchi dichiarazioni consentiti si rimanda al capitolo 8.4. Si deve tener presente che la parte dichiarazioni determina anche la struttura del DB d'istanza assegnato.

Esempi:

```
CONST
   COSTANTE:=5;
END_CONST


VAR
   VALORE1,VALORE2,VALORE3:INT;
END_VAR
```

Esempio

L’esempio 8-2 illustra il codice sorgente per un blocco funzionale. In questo caso i parametri di ingresso e uscita (qui V1, V2) sono stati impostati con valori iniziali.

```
FUNCTION_BLOCK FB11
      VAR_INPUT
            V1: INT:= 7;
      END_VAR
      VAR_OUTPUT
            V2: REAL;
      END_VAR
      VAR
            LOOP_1:INT;
      END_VAR
      BEGIN
      IF V1 = 7 THEN
      LOOP_1:= V1;
      V2:= FC2 (VALORETEST:= LOOP_1);
      //Richiamo della funzione FC2 con
      //assegnazione parametri tramite la
      //variabile LOOP_1
      END_IF;
END_FUNCTION_BLOCK
```

Esempio 8-2 Sintassi di un blocco funzionale

8.8 Struttura di una funzione (FC)

Panoramica

Una funzione (FC) è un blocco di codice al quale non è assegnata alcuna area di memoria. Essa non necessita di alcun DB di istanza. Contrariamente ad un FB, tale funzione può ritornare un risultato di funzione (**valore di ritorno**) al punto di richiamo. Perciò, tale funzione può essere utilizzata come una variabile in un'espressione. Le funzioni del tipo VOID non hanno alcun valore di ritorno.

Funzione

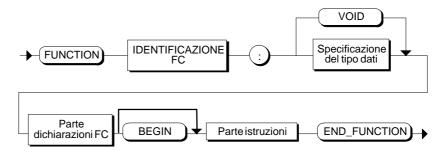



Figura 8-11 Sintassi di una funzione FC

Identificazione FC

Dopo la parola chiave

```
FUNCTION
```

introdurre come nome della FC la parola chiave FC seguita dal numero del blocco oppure il nome simbolico dell'FC.

Esempi:

```
FUNCTION FC100
FUNCTION NUMERO DI GIRI
```

Specificazione del tipo di dati

Qui si deve indicare il tipo di dati del valore di ritorno. Sono consentiti tutti i tipi di dati descritti nel capitolo 9, fatta eccezione per i tipi di dati STRUCT e ARRAY. L'indicazione di un tipo di dati può essere omessa se con **VOID** si rinuncia al valore di ritorno.

Parte dichiarazioni FC

Per i blocchi dichiarazioni consentiti si rimanda al capitolo 8.4.

Parte istruzioni

Nell'ambito della parte istruzioni, al nome della funzione si deve assegnare il **risultato della funzione**. Segue un esempio di istruzione valida nell'ambito di una funzione con la denominazione FC31, p. es.:

```
FC31:= VALORE;
```

Esempio

L'esempio riportato illustra una funzione con i parametri d'ingresso formali x1, x2, y1, y2, con una parametro di uscita formale Q2 e con un valore di ritorno FC11.

Per il significato dei parametri formali si rimanda al capitolo 10.

```
FUNCTION FC11: REAL
      VAR_INPUT
            x1: REAL;
            x2: REAL;
            y1: REAL;
            y2: REAL;
      END_VAR
      VAR_OUTPUT
            Q2: REAL;
      END_VAR
      BEGIN       // Parte istruzioni
      FC11:= SQRT // Ritorno del valore della
                  // funzione
      ( (x2 - x1)**2 + (y2 - y1) **2 );
      Q2:= x1;
END_FUNCTION
```

Esempio 8-3 Esempio di una funzione

8.9 Struttura di un blocco organizzativo (OB)

Panoramica

Il blocco organizzativo (OB), come un FB o una FC, fa parte del programma utente e viene richiamato dal sistema operativo a intervalli ciclici o quando si verificano determinati eventi. Esso costituisce l'interfaccia fra programma utente e sistema operativo.

Blocco organizzativo

Figura 8-12 Sintassi di un blocco organizzativo

Identificazione OB

Dopo la parola chiave

```
ORGANIZATION_BLOCK
```

introdurre come nome dell'OB la parola chiave OB seguita dal numero del blocco oppure il nome simbolico dell'OB.

Esempi:

```
ORGANIZATION_BLOCK OB14
ORGANIZATION_BLOCK ALLARME DALL'OROLOGIO
```

Parte dichiarazioni OB

In linea di massima, ogni OB necessita di **20 byte di dati locali** per l'informazione di avvio. A seconda delle esigenze del programma, nell'OB si possono dichiarare ulteriori variabili temporanee. Per la descrizione dei 20 byte di dati locali si rimanda a **/235/**.

Esempio:

```
ORGANIZATION_BLOCK OB14
// ALLARME DALL'OROLOGIO

VAR_TEMP
HEADER:ARRAY [1..20] OF BYTE;// 20 Byte per informazione
di avvio
:
:
END_VAR
```

Per i rimanenti blocchi dichiarazioni consentiti per OB si rimanda al capitolo 8.4.

8.10 Struttura di un blocco dati (DB)

Panoramica

Il blocco dati DB contiene dati globali specifici dell'utente ai quali possono accedere **tutti** i blocchi del programma. Ogni FB, FC o OB può leggere o scrivere in questo DB. La struttura dei blocchi dati, ai quali sono assegnati solo determinati FB, (i DB d'istanza) sono descritti nel capitolo 12.

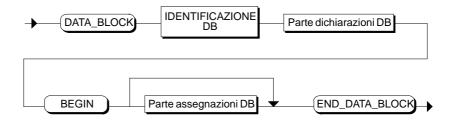



Figura 8-13 Sintassi di un blocco dati (DB)

Identificazione del DB

Dopo la parola chiave

```
DATA_BLOCK
```

introdurre come nome del DB la parola chiave DB seguita dal numero del blocco oppure il nome simbolico del DB.

Esempio:

```
DATA_BLOCK DB20
```

```
DATA_BLOCK CAMPO DI MISURA
```

Parte dichiarazioni DB

Nella parte dichiarazioni del DB viene definita la struttura dati del DB. Ad una variabile DB può essere assegnato un tipo di dati strutturato (STRUCT) o un tipo di dati definito dall'utente (UDT).

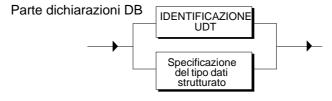



Figura 8-14 Sintassi della parte dichiarazioni DB

Esempio:

```
DATA_BLOCK DB20
            STRUCT      // Parte dichiarazioni
              VALORE:ARRAY [1..100] OF INT;
            END_STRUCT

BEGIN                   // Inizio parte assegnazioni
:
:

END_DATA_BLOCK          // Fine Blocco dati
```

Parte assegnazioni DB

Nella parte assegnazioni, i dati che sono stati definiti nella parte dichiarazioni possono essere adattati per applicazioni specifiche con singoli valori specifici del DB. La parte assegnazioni inizia con la parola chiave:

BEGIN

e si compone quindi di una sequenza di assegnazioni di valori.

Parte assegnazioni DB

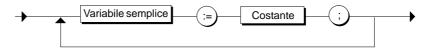



Figura 8-15 Parte assegnazioni di un DB globale

Avvertenza

Quando si assegnano dei valori iniziali (inizializzazione), si indicano degli attributi e dei commenti all'interno di un DB, si utilizzi la sintassi di AWL. Per informazioni su come scrivere costanti, attributi e commenti, consultare il manuale utente **/231/** o il manuale **/232/**.

Esempio

L'esempio seguente illustra in che modo può essere formulata la parte assegnazioni quando i valori del campo [1] e [5] devono avere il numero intero 5 e –1 invece della predefinizione 1.

```
DATA_BLOCK  DB20
STRUCT              //Dichiarazione dati con
                    //predefinizione
      VALORE        : ARRAY [ 1..100] OF INT := 100
(1);
      MERKER:BOOL   := TRUE;
      S_WORT:WORD   := W#16#FFAA;
      S_BYTE:BYTE   := B#16#FF;
      S_TIME:S5TIME := S5T#1h30m30s;
END_STRUCT

BEGIN              //Parte assegnazioni
   //Assegnazione di valori per determinati
   //elementi di campo
   VALORE [1]      := 5;
   VALORE [5]      :=–1;
END_DATA_BLOCK
```

Esempio 8-4 Parte assegnazioni di un DB

8.11 Struttura di un tipo di dati definito dall’utente (UDT)

Panoramica

I tipi di dati definiti dall’utente (UDT) sono strutture speciali di dati generati dall’utente. Poiché i tipi di dati definiti dall’utente possiedono un nome, essi possono essere impiegati più volte. Dopo la loro definizione, essi possono essere utilizzati nell’intero programma utente e sono quindi tipi di dati globali. Perciò, questi tipi di dati possono essere utilizzati:

- nel blocco come tipi di dati semplici o tipi di dati composti oppure
- come modello per la creazione di blocchi dati con la stessa struttura.

Tipo di dati definito dall’utente

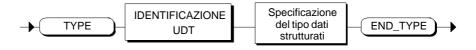



Figura 8-16 Sintassi per il tipo di dati definito dall’utente (UDT)

Identificazione UDT

Dopo la parola chiave

```
TYPE
```

introdurre la parola chiave UDT seguita da un numero o semplicemente dal nome simbolico dell’UDT.

Esempi:

```
TYPE UDT 10
TIPO BLOCCO ALIMENTAZIONE
```

Specificazione dei tipi di dati

La specificazione del tipo di dati avviene sempre con l’ausilio di una **specificazione di tipi di dati STRUCT**. Il tipo di dati può essere utilizzato risp. assegnato nei blocchi dichiarazione di blocchi di codice oppure in blocchi dati risp. DB. Per i blocchi dichiarazione consentiti e ulteriori informazioni si rimanda al capitolo 9.

Tipi di dati 9

Panoramica

Un tipo di dati è la riunione di campi di valori e operazioni per formare delle unità. Come la maggior parte dei linguaggi di programmazione, SCL possiede tipi di dati predefiniti (ossia integrati nel linguaggio). Inoltre, il programmatore può creare tipi di dati composti e definiti dall'utente.

Sommario del capitolo

9.1 Panoramica

Panoramica

Nell'ambito di SCL, si distinguono i vari tipi di dati secondo la tabella 9-1:

Tabella 9-1 Tipi di dati in SCL

Tipi di dati semplici			
BOOL	CHAR	INT	TIME
BYTE		DINT	DATE
WORD		REAL	TIME_OF_DAY
DWORD			S5TIME
Tipi di dati composti			
DATE_AND_TIME	STRING	ARRAY	STRUCT
Tipi di dati definiti dall'utente			
UDT			
Tipi di parametri			
TIMER	BLOCK_FB	POINTER	ANY
COUNTER	BLOCK_FC		
	BLOCK_DB		
	BLOCK_SDB		

Questi tipi di dati determinano:

- il tipo e il significato degli elementi di dati
- i campi consentiti degli elementi di dati
- la quantità consentita di operazioni che possono essere eseguite con un operando di un tipo di dati
- il modo di scrittura dei dati di questo tipo

9.2 Tipi di dati semplici

Panoramica

I tipi di dati semplici definiscono la struttura di dati che non possono essere suddivisi in unità inferiori. Essi corrispondono alla definizione della norma DIN EN 1131-3. Un tipo di dati semplice descrive un'area di memoria di lunghezza fissa e indica numero di bit, integer, real, durata, ora dimensione dei caratteri. Tutti questi tipi di dati sono predefiniti in SCL.

Tabella 9-2 Numeri di bit e campi di valori dei tipi di dati semplici

Tipo	Parola chiave	Numero di bit	Campo di valori
Tipo di dati di bit	I dati di questo tipo occupano 1 bit (tipo di dati BOOL), 8 bit, 16 bit o 32 bit		
Bit	BOOL	1	0,1 oppure FALSE, TRUE
Byte	BYTE	8	Non può essere indicato un campo di valori numerici. Si tratta di una combinazione di bit con i quali non si possono generare espressioni numeriche.
Parola	WORD	16	
Doppia parola	DWORD	32	
Tipo di carattere	I dati di questo tipo occupano esattamente 1 carattere del set di caratteri ASCII		
Carattere singolo	CHAR	8	set di caratteri ASCII ampliato
Tipo di dati numerico	Sono disponibili per l'elaborazione di valori numerici.		
Numero intero	INT	16	-32_768 fino a 32_767
Doppio numero intero	DINT	32	-2_147_483_648 fino a 2_147_483_647
Numero in virgola mobile (numero in virgola mobile IEE)	REAL	32	-3.402822E+38 fino a -1.175495E-38, 0.0, +1.175495E-38 fino a 3.402822E+38
Tipo di temporizzatore	I dati di questo tipo rappresentano svariati valori di tempo/data nell'ambito di STEP 7.		
Temporizzatore S5	S5TIME	16	T#0H_0M_0S_10MS fino a T#2H_46M_30S
Dati dei temporizzatori: Tempo IEC a intervalli di 1 ms	TIME (=DURATION)	32	-T#24D_20H_31M_23S_647MS fino a T#24D_20H_31M_23S_647MS
Data: Data IEC a intervalli di 1 giorno	DATE	16	D#1990-01-01 fino a D#2168-12-31
Ora del giorno: Ora ad intervalli di 1 ms	TIME_OF_DAY (=TOD)	32	TOD#0:0:0 fino a TOD#23:59:59.999

Nota per il temporizzatore S5: a seconda della base di tempo – 0.01S, 0.1S, 1S o 10S – la risoluzione del valore di tempo è limitata. Il Compilatore arrotonda i valori in modo opportuno.

9.3 Tipi di dati composti

Panoramica

SCL supporta i seguenti tipi di dati composti:

Tabella 9-3 Tipi di dati composti

Tipo di dati	Descrizione
DATE_AND_TIME DT	Definisce un campo di 64 bit (8 byte). Questo tipo di dati memorizza (in formato decimale in codice binario) data e ora ed è già predefinito in SCL.
STRING	Definisce un campo per una sequenza di max. 254 caratteri (tipo di dati CHAR).
ARRAY	Definisce un campo con elementi di un tipo di dati (semplici o composti).
STRUCT	Definisce un raggruppamento di tipi di dati con qualunque combinazione. Si possono definire campi di strutture oppure strutture di strutture e campi.

9.3.1 Tipo di dati DATE_AND_TIME

Panoramica

Il tipo di dati DATE_AND_TIME è composto dai tipi di dati DATE e TIME. Esso definisce un'area di 64 bit (8 byte) per l'indicazione di data e ora. Nell'area di dati vengono memorizzate le seguenti informazioni: anno, mese, giorno, ore, minuti, secondi, millisecondi.

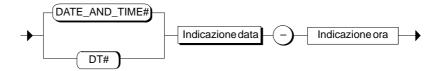



Figura 9-1 Sintassi di DATE_AND_TIME

Tabella 9-4 Numero di bit e campo di valori

Campo di valori

Tipo	Parola chiave	Numero di bit	Area di valori
Data e ora	DATE_AND_TIME (=DT)	64	DT#1990-01-01:0:0:0.0 fino a DT#2089-12-31:23:59:59.999

Una descrizione dettagliata della sintassi per l'indicazione della data e dell'ora si trova nel capitolo 11 di questo manuale. Una definizione valida per il 20.10.1995 alle ore 12.20 min. 30 sec. e 10 millisecondi è la seguente:

```
DATE_AND_TIME#1995-10-20-12:20:30.10
```

```
DT#1995-10-20-12:20:30.10
```

Avvertenza

Per accedere direttamente ai componenti DATE o TIME sono disponibili FC standard.

9.3.2 Tipo di dati STRING

Panoramica

Un tipo di dati STRING definisce una stringa di caratteri di max. 254 caratteri singoli.

L'area standard riservata per una stringa di caratteri si compone di 256 byte. Quest'area di memoria è necessaria per memorizzare 254 caratteri e un'intestazione di 2 byte.

Si può ridurre lo spazio necessario per una stringa di caratteri definendo il numero max. di caratteri che devono essere memorizzati nella stringa. Una *stringa zero*, ossia una stringa senza contenuto, rappresenta il valore minimo possibile.

Specificazione del tipo dati STRING

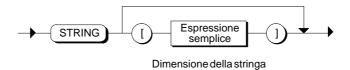



Figura 9-2 Sintassi della specificazione del tipo dato STRING

L'espressione semplice indica il numero max. di caratteri nella stringa di caratteri.

Seguono alcuni tipi di STRING validi:

```
STRING[10]
STRING[3+4]
STRING[3+4*5]
STRING      campo di valori max. (default
```
254 $\triangleq$ caratteri)

Campo di valori

In una stringa di caratteri sono consentiti tutti i caratteri del codice ASCII. Il capitolo 11 descrive come trattare i caratteri di controllo e i caratteri non stampabili.

Avvertenza

L'area di 254 caratteri riservata nella versione standard può essere ridotta a un numero di caratteri a scelta per poter meglio usufruire delle risorse della CPU. Selezionare il comando di menu **Impostazioni** nel menu **Strumenti** e la scheda ”Compilatore” nella finestra di dialogo successiva. Immettere quindi nell'opzione ”Lunghezza massima della stringa” il numero desiderato di caratteri.

9.3.3 Tipo di dati ARRAY

Panoramica

Il tipo di dati ARRAY ha un numero fisso di componenti di un solo tipo di dati. Nel diagramma sintattico 9-3 per ARRAY, questo tipo di dati viene specificato più dettagliatamente dopo la parola riservata `OF.` SCL distingue:

- il tipo ARRAY a una dimensione
  (Lista di elementi di dati ordinati in ordine crescente)
- il tipo ARRAY a due dimensioni
  (Tabella di dati composta di righe e colonne. La prima dimensione si riferisce al numero di riga e la seconda al numero di colonna)
- il tipo di ARRAY di dimensione superiore
  (Estensione del tipo di ARRAY a due dimensioni che viene ampliato con ulteriori dimensioni. Il numero max. di dimensioni consentite è 6).

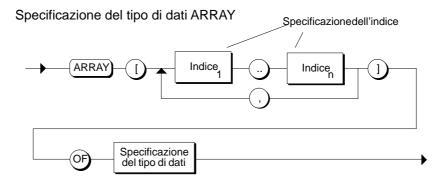



Figura 9-3 Sintassi della specificazione del tipo di dati ARRAY

Specificazione dell'indice

Vengono descritte le dimensioni del campo con

- l'indice minimo e massimo possibile (campo indice) per ogni dimensione. L'indice può essere un numero intero qualsiasi (da -32768 fino a 32767).
- i limiti devono essere separati da due punti.
- Le singole aree di indice devono essere separate mediante virgole. L'intera specificazione indice viene racchiusa fra parentesi quadre.

Specificazione del tipo dati

Con la specificazione del tipo di dati si dichiara il tipo di dati dei componenti. Sono consentiti tutti i tipi di dati elencati in questo capitolo. Il tipo di dati di un ARRAY può a sua volta essere una struttura.

Le seguenti specificazioni sono possibili tipi di ARRAY:

```
ARRAY[1..10] OF REAL
ARRAY[1..10] OF STRUCT..END_STRUCT
ARRAY[1..100, 1..10] OF REAL
```

9.3.4 Tipo di dati STRUCT

Panoramica

Un tipo di dati STRUCT descrive un'area composta da un numero fisso di componenti, i quali possono essere di qualsiasi tipo. Nel diagramma sintattico 9-4 questi elementi di dati vengono indicati subito dopo la parola chiave STRUCT nella dichiarazione dei componenti.

In particolare, un elemento di dati del tipo STRUCT può essere a sua volta del tipo strutturato. In altre parole: è consentito l'annidamento del tipo di dati STRUCT.

Nel capitolo 10 viene descritto in che modo si può accedere ai dati di una struttura.

Specificazione del tipo di dati STRUCT

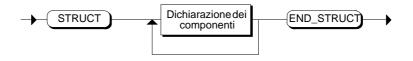



Figura 9-4 Sintassi della specificazione del tipo di dati STRUCT

Dichiarazione dei componenti

Questo è un elenco dei vari componenti di una struttura. In base al diagramma sintattico 9-5 questo elenco si compone

- di almeno 1 fino a n identificatori
- del corrispondente tipo di dati
- di un'impostazione opzionale con valori iniziali.

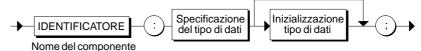
Dichiarazione dei componenti

Nomi dei componenti nell'ambito di strutture

Figura 9-5 Sintassi della dichiarazione dei componenti per il tipo di dati STRUCT

Identificatore

Si tratta del nome di un elemento di struttura per la seguente specificazione di dati.

Inizializzazione del tipo di dati

Come opzione, un singolo elemento della struttura può essere impostato con un valore iniziale conformemente alla specificazione dei dati. Questa assegnazione avviene tramite un’assegnazione di valori e viene descritta nel capitolo 10.

Esempio

Nell’esempio viene descritta una possibile definizione di un tipo di dati `STRUCT`.

```
STRUCT
 //INIZIO Dichiarazione componenti
  A1        :INT;
  A2        :STRING[254];
  A3        :ARRAY [1..12] OF REAL;
  |
  Nomi dei componenti  | Specificazioni dei dati
//ENDE Dichiarazione componenti
END_STRUCT
```

Esempio 9-1 Definizione di un tipo di dati `STRUCT`

9.4 Tipo di dati definito dall’utente (UDT)

Panoramica

Un tipo di dati UDT viene definito come blocco (vedere capitolo 8). Un tale tipo di dati, in base alla sua definizione può essere utilizzato nell’intero programma utente e è quindi un tipo di dati globale. Questi tipi di dati possono essere utilizzati con la loro identificazione UDT UDTx (x indica il numero) oppure con un nome simbolico assegnato nella parte dichiarazioni di un blocco o di un blocco dati.

Tipo di dati definito dall’utente

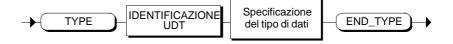



Figura 9-6 Sintassi del tipo di dati definito dall’utente (UDT)

Identificazione UDT

Una dichiarazione di un UDT viene indicata dalla parola chiave TYPE, seguita dal nome dell’UDT (identificatore UDT). Conformemente al capitolo 8 il nome dell’UDT può essere assoluto, cioè indicato con un nome standard in forma UDTx (x è in numero), oppure può essere indicato con un nome simbolico.

Specificazione del tipo di dati

Dopo l’identificazione UDT segue la specificazione del tipo di dati. Qui è consentita solo la specificazione del tipo di dati STRUCT (vedere capitolo 9.3.4):

```
STRUCT
:
END_STRUCT
```

Dopodiché, l’intera dichiarazione di un UDT viene conclusa con la parola chiave

```
END_TYPE
```

Impiego di un UDT

Il tipo di dati così definito può essere impiegato per l’uso di variabili o parametri nonché per la dichiarazione di DB. Con l’ausilio dell’UDT si possono anche definire componenti di strutture o campi, anche nell’ambito di altri UDT.

Avvertenza

Nell’assegnazione di valori iniziali (inizializzazione) all’interno di un UDT vale la sintassi di AWL. Per informazioni su come scrivere le costanti, consultare il manuale utente **/231/** o il manuale **/232/**.

Esempio

L'esempio descrive una definizione di un UDT e l'impiego di questo tipo di dati nell'ambito di una dichiarazione di variabili. Si presuppone che nella tabella dei simboli sia stato definito il nome ”VALORIMISURA” per l'UDT50.

```
TYPE VALORIMISURA // Definizione dell'UDT
STRUCT
  BIPOL_1 : INT;
  BIPOL_2 : WORD   := W#16#AFA1;
  BIPOL_3 : BYTE   := B#16#FF;
  BIPOL_4 : WORD   := B#(25,25);
  BIPOL_5 : INT    := 25;
MISURA:          STRUCT
                 BIPOLAR_10V: REAL;
                 UNIPOLAR_4_20MA: REAL;
                 END_STRUCT;
END_STRUCT
END_TYPE
```

```
FUNCTION_BLOCK
VAR
      CAMPO_MISURA: VALORIMISURA;
END_VAR
BEGIN
...
      CAMPO_MISURA.BIPOL:= -4;
      CAMPO_MISURA.MISURA.UNIPOLAR_4_20MA:= 2.7;
...
END_FUNCTION_BLOCK;
```

Esempio 9-2 Convenzione di tipi di dati definiti dall'utente

9.5 Tipi di parametri

Panoramica

Oltre ai tipi di dati semplici, composti e definiti dall'utente, per la definizione di parametri formali del blocco degli FB e FC si possono utilizzare dei cosiddetti **tipi di parametri**. Questi tipi di dati servono per

- definire come parametri funzioni di contatori e temporizzatori (TIMER/COUNTER),
- definire FC, FB, DB e SDB come parametri (BLOCK_xx)
- consentire come parametro un operando di qualsiasi tipo di dati (ANY)
- consentire come parametro un'area di memoria (POINTER)

Tabella 9-5 Tipo di parametri

Parametro	Dimensione	Descrizione
TIMER	2 byte	Contrassegna un determinato timporizzatore che viene utilizzato dal programma nel blocco di codice richiamato. Parametro attuale: p. es. T1
COUNTER	2 byte	Contrassegna un determinato contatore che viene utilizzato dal programma nel blocco codice richiamato. Parametro attuale: p. es. Z10
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	2 byte	Contrassegna un determinato blocco che viene utilizzato dal programma nel blocco codice richiamato. Parametro attuale: p. es. FC101 DB42
ANY	10 byte	Viene impiegato nei casi in cui come tipo di dati del parametro attuale è consentito un tipo di dati qualsiasi.
POINTER	6 byte	Contrassegna una determinata area di memoria che deve essere impiegata dal programma. Parametro attuale: p. es. M50.0

TIMER e COUNTER

Viene definito un determinato temporizzatore o contatore, che si desidera utilizzare per l'elaborazione di un blocco. I tipi di dati TIMER e COUNTER sono consentiti solo per parametri d'ingresso (VAR_INPUT).

Tipi di BLOCK

Viene definito un determinato blocco, che si desidera utilizzare come parametro d'ingresso. La dichiarazione del parametro d'ingresso determina il tipo di blocco (FB, FC, DB). Durante l'assegnazione dei parametri viene indicato il nome del blocco. Sono consentiti sia l'indicatore assoluto (p. es. FB20) che l'indicatore simbolico.

SCL non offre alcuna operazione per questo tipo di dati. Si possono solo assegnare parametri di questo dipo durante il richiamo di blocchi. Nelle FC non è consentito il trasferimento di un parametro d'ingresso.

In SCL si possono assegnare come parametri attuali gli operandi dei seguenti tipi di dati:

- blocchi funzionali senza parametri formali
- funzioni senza parametri formali e valore di ritorno (VOID)
- blocchi dati e blocchi dati di sistema.

Tipo di dati ANY

SCL offre la possibilità di definire parametri di blocco del tipo di dati ANY; durante il richiamo di un simile tipo di blocco, questi parametri possono essere alimentati con operandi di qualsiasi tipo di dati. SCL offre tuttavia solo una possibilità di elaborazione del tipo di dati ANY: il trasferimento a blocchi subordinati.

In SCL, si possono assegnare come parametri attuali gli operandi dei seguenti tipi di dati:

- Tipi di dati semplici:
  si deve indicare l'indirizzo assoluto o il nome simbolico del parametro attuale.
- Tipi di dati composti:
  si deve indicare il nome simbolico dei dati di tipo composto (p. es. campi e strutture).
- Tipo di dati ANY:
  esso è possibile solo se l'operando è un paramero formale di tipo di parametro compatibile.
- Tipo di dati NIL:
  viene specificato un puntatore zero.
- Temporizzatori, contatori e blocchi:
  si deve indicare la relativa identificazione (p. es. T1, Z20 o FB6).

Il tipo di dati ANY è consentito per parametri d'ingresso formali, per parametri di transito di FB e FC e per parametri di uscita di FC.

Avvertenza

Se nel richiamare un FB o una FC al parametro formale del tipo ANY viene assegnata una variabile temporanea, questo parametro non può essere trasferito ad un altro blocco nel blocco richiamato. Gli indirizzi delle variabili temporanee perdono la loro validità al momento del trasferimento.

POINTER

In SCL esiste la possibilità di impostare parametri di blocco del tipo di dati `POINTER`, ove al richiamo di tale blocco a questi parametri possono essere assegnati operandi di tipo di dati a piacere. SCL offre tuttavia solo una possibilità di elaborazione del tipo di dato `POINTER`: il trasferimento a blocchi subordinati.

In SCL è possibile assegnare come parametri gli operandi del tipo di dati seguenti:

- Tipi di dati semplici:
  Introdurre l'indirizzo assoluto o il nome simbolico del parametro attuale.
- Tipi di dati composti:
  Introdurre il nome simbolico dei dati di tipo composto (ad es. campi e strutture).
- Tipo di dati `POINTER`:
  Ciò è possibile solo se l'operando è un parametro formale di tipo compatibile.
- Tipo di dati `NIL`:
  Introdurre un puntatore zero

Il tipo di dati `POINTER` è consentito per i parametri di ingresso, i parametri di transito di FB e FC e per i parametri di uscita di FC.

Avvertenza

Se nel richiamare un FB o una FC a un parametro formale del tipo `POINTER` viene assegnata una variabile temporanea, nel blocco richiamato questo parametro non può essere trasferito ad un altro blocco. Gli indirizzi delle variabili temporanee perdono la loro validità al momento del trasferimento.

Esempi

```
FUNCTION ABSTAND: REAL
      VAR_INPUT
            MioDB:BLOCK_DB;
            TEMPO : TIMER;
      END_VAR
      VAR
            INDEX: INTEGER;
      END_VAR
      BEGIN
      MioDB.DB5:=5;
      DISTANZA:=....    // VALORERITORNO
END_FUNCTION
```

Esempio 9-3 Tipo di dati BLOCK_DB e TIMER

```
FUNCTION FC100: VOID
      VAR_IN_OUT
            in, out:ANY;
      END_VAR
      VAR_TEMP
            ret: INT;
      END_VAR
      BEGIN
      //...
      ret:=SFC20(DSTBLK:=out,SCRBLK:=in);
      //...
END_FUNCTION

FUNCTION_BLOCK FB100
      VAR
            ii:INT;
            aa, bb:ARRAY[1..1000] OF REAL;
      END_VAR
      BEGIN
      //...
      FC100(in:=aa, out:=bb);
      //...
END_FUNCTION_BLOCK
```

Esempio 9-4 Tipo di dati ANY

Definizione di variabili locali e parametri di blocco 10

Panoramica

Le variabili locali e i parametri di blocco sono dati che vengono definiti nell'ambito di un blocco di codice (FC, FB, OB) e che sono validi solo per questo blocco di codice. Il presente capitolo contiene informazioni sulla definizione e l'inizializzazione di questi dati.

Sommario del capitolo

10.1 Panoramica

Suddivisione delle variabili

Le variabili locali possono essere suddivise nelle seguenti categorie conformemente alla tabella 10-1:

Tabella 10-1 Variabili locali

Variabile	Significato
Variabili statiche	Le variabili statiche sono variabili locali il cui valore rimane immutato ad ogni esecuzione del blocco (memoria del blocco). Esse servono per memorizzare i valori di un blocco funzionale e vengono depositate nel blocco dati di istanza.
Variabili temporanee	Le variabili temporanee fanno parte di un blocco di codice e non occupano **alcuna** area di memoria statica poiché esse vengono deposte nello stack della CPU. Il loro valore rimane immutato solo per la durata di una esecuzione del blocco. Al di fuori del blocco in cui sono state dichiarate le variabili **non** è possibile accedere alle variabili temporanee.

Suddivisione dei parametri di blocco

I parametri di blocco sono dei segnaposti che vengono definiti solo in caso di utilizzo effettivo (richiamo) del blocco. I segnaposti presenti nel blocco vengono denominati parametri formali, mentre i valori assegnati durante il richiamo del blocco vengono denominati parametri attuali. I parametri formali di un blocco possono essere considerati come variabili locali.

Conformemente alla tabella 10-2, i parametri di blocco possono essere suddivisi nelle seguenti categorie:

Tabella 10-2 Parametri di blocco

Parametri di blocco	Significato
Parametri d'ingresso	I parametri d'ingresso assumono i valori d'ingresso attuali durante il richiamo del blocco. Essi possono essere solo letti.
Parametri d'uscita	I parametri d'uscita trasferiscono i valori d'uscita attuali al blocco richiamante. Essi possono essere letti e scritti.
Parametri di transito	Al momento del richiamo del blocco, i parametri di transito assumono il valore attuale di una variabile, lo elaborano e infine depositano i risultati ottenuti nella stessa variabile.

Flag (Flag OK)

Il compilatore SCL mette a disposizione un flag che consente di localizzare eventuali errori durante l'esecuzione dei programmi nella CPU. Si tratta di una variabile locale del tipo BOOL con il nome predefinito ”OK”.

Dichiarazione di variabili e parametri

Conformemente alla tabella 10-3, ad ogni categoria di variabili o parametri locali viene assegnato un proprio blocco convenzioni il quale a sua volta viene contrassegnato da una propria coppia di parole chiavi.

Ogni blocco contiene le dichiarazioni consentite per questo blocco dichiarazioni. Esso può essere presente una sola volta nella parte dichiarazioni del blocco, ma la sequenza di questi blocchi non è rilevante.

Nella tabella 10-3, i blocchi convenzione consentiti in un blocco sono contrassegnati con una ”x”.

Tabella 10-3 Blocchi dichiarazioni per variabili locali e parametri

Dati	Sintassi	FB	FC	OB
Variabili statiche	VAR : END_VAR	X	X[1]	
Variabili temporanee	VAR_TEMP : END_VAR	X	X	X
Parametri di blocco come: parametri d’ingresso	VAR_INPUT : END_VAR	X	X	
parametri d’uscita	VAR_OUTPUT : END_VAR	X	X	
parametri di transito	VAR_IN_OUT : END_VAR	X	X	

[1] *La dichiarazione di variabili all’interno della coppia di parole chiave VAR e END_VAR è ammessa nelle funzioni ma con la compilazione le dichiarazioni vengono spostate nell’area temporanea.*

Inizializzazione

In fase di dichiarazione, alle variabili ed ai parametri occorre assegnare un tipo di dati che determina la struttura e quindi anche la memoria necessaria. Inoltre, alle variabili statiche ed ai parametri di un blocco funzionale si possono assegnare determinati valori iniziali. La tabella 10-4 contiene una panoramica indicante in quali casi è possibile una inizializzazione.

Tabella 10-4 Inizializzazione di dati locali

Categoria di dati	Inizializzazione
Variabili statiche	possibile
Variabili temporanee	non possibile
Parametri di blocco	possibile solo per i parametri d’ingresso e d’uscita di un blocco funzionale

10.2 Dichiarazione di variabili e parametri

Panoramica

Una dichiarazione di variabili o parametri si compone di un identificatore liberamente scelto per il nome della variabile e per l'indicazione del tipo di dati. Il diagramma sintattico illustra la forma generale. Il capitolo 8.4 contiene una descrizione dettagliata dell'assegnazione di attributi di sistema per parametri.

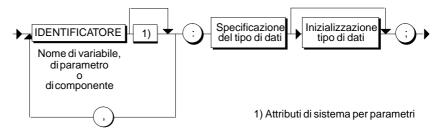



Figura 10-1 Sintassi della dichiarazione di variabili

Seguono alcuni esempi di dichiarazioni valide:

```
VALORE1:  REAL;
```

oppure se esistono diverse variabili dello stesso tipo:

```
VALORE2, VALORE 3, VALORE4,....: INT;

CAMPO  :  ARRAY[1..100, 1..10] OF REAL;

RECORD :  STRUCT
             CAMPOMISURA:ARRAY[1..20] OF REAL;
             INTERRUTTORE:BOOL;
          END_STRUCT
```

Specificazione del tipo di dati

Sono consentiti tutti i tipi di dati trattati nel capitolo 9.

Avvertenza

Si possono dichiarare come identificatori parole riservate che hanno validità solo in SCL, preponendo il carattere ”#” (p. es. #FOR). Ciò può essere vantaggioso quando si vogliono trasmettere dei parametri attuali a blocchi che sono stati creati in un altro linguaggio (p. es. AWL).

10.3 Inizializzazione

Principio

Nella dichiarazione, le variabili statiche, i parametri d'ingresso di un FB e i parametri di uscita di un FB possono essere impostati con un determinato valore. Questa impostazione iniziale avviene con un'assegnazione di valore (:=) dopo l'indicazione del tipo di dati. Conformemente al diagramma sintattico 10-2 si può assegnare:

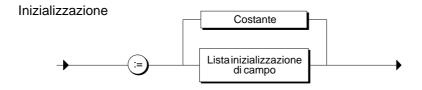
- una costante ad una variabile semplice oppure
- una lista di inizializzazione ad un campo.

Inizializzazione

Figura 10-2 Sintassi dell'inizializzazione tipo dati

Esempio:

```
VALORE :REAL  :=  20.25;
```

L'inizializzazione di una lista di variabili (A1, A2, A3,...: INT:=...) non è possibile. In tal caso le variabili devono essere inizializzate singolarmente. I campi vengono impostati in base alla figura 10-3 con dei valori iniziali.

Lista inizializzazione campo

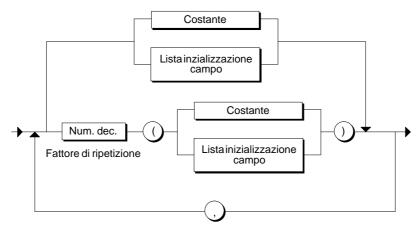
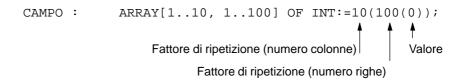



Figura 10-3 Sintassi dell'inizializzazione di campi

```
CAMPO :     ARRAY[1..10, 1..100] OF INT:=10(100(0));
```

Esempi:

L'esempio 10-1 illustra l'inizializzazione di una variabile statica:

```
VAR
      INDEX1: INT:= 3;
END_VAR
```

Esempio 10-1 Inizializzazione di variabili statiche

L'esempio 10-2 illustra l'inizializzazione di un campo a due dimensioni. Se si desidera definire in SCL la seguente struttura di dati sotto il nome REGOLATORE, si deve scrivere:

-54	736	-83	77
-1289	10362	385	2
60	-37	-7	103
60	60	60	60

```
VAR
REGOLATORE:
ARRAY [1..3, 1..4] OF INT:=-54,  736, -83, 77,
                        -1289, 10362,  385, 2,
                           60,  -37,   -7, 103,
                        4(60);

END_VAR
```

Esempio 10-2 Inizializzazione di campo

L'esempio 10-3 illustra l'inizializzazione di una struttura:

```
VAR
GENERATORE:STRUCT
  DATI:  REAL     := 100.5;
  A1:    INT      := 10;
  A2:    STRING[6]:= 'FATTORE';
  A3:    ARRAY[1..12] OF REAL:= 12(100.0);
END_STRUCT
END_VAR
```

Esempio 10-3 Inizializzazione di struttura

10.4 Dichiarazione di istanze

Panoramica

Nella parte dichiarazioni di blocchi funzionali, oltre alle variabili già note con tipi di dati semplici, composti o definiti dall'utente, si possono definire anche variabili di tipo FB o SFB. Queste variabili vengono denominate **istanze locali** dell'FB o dell'SFB.

I dati di istanza locali vengono memorizzati nel blocco dati di istanza del blocco funzionale richiamante.

Figura 10-4 Sintassi della dichiarazione di istanze locali

Esempi: seguono alcuni esempi validi conformemente alla sintassi della figura 10-4:

```
Alimentazione1          : FB10;
Alimentazione2, Alimentazione3, Alimentazione4 : FB100;
Motore1                 : Motore ;
```

”// Motore” è un simbolo registrato nella tabella dei simboli.

Simbolo	Indirizzo	Tipo di dati
MOTORE	FB20	FB20

Figura 10-5 Corrispondente tabella dei simboli in STEP 7

Inizializzazione

Un'inizializzazione locale, specifica dell'istanza non è possibile.

10.5 Variabili statiche

Panoramica

Le variabili statiche sono variabili locali i cui valori rimangono inalterati in tutte le esecuzioni dei blocchi (memoria dei blocchi). Esse servono per memorizzare il valore di un blocco funzionale e vengono memorizzate nel corrispondente blocco dati di istanza.

Blocco di variabili statiche

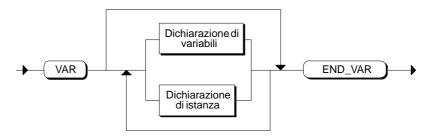



Figura 10-6 Sintassi del blocco variabili statiche

Blocco dichiarazioni
VAR
END_VAR

Questo blocco dichiarazioni fa parte della parte dichiarazioni dell'FB. In questo blocco si possono:

- definire nomi di variabili e tipi di dati con la dichiarazione di variabili secondo il capitolo 10.2, come opzione anche con inizializzazione
- inserire altre dichiarazioni di variabili già esistenti con la dichiarazione di istanze secondo il capitolo 10.4.

Dopo la compilazione, insieme con i blocchi per i parametri del blocco, questo blocco determina la struttura del blocco dati di istanza assegnato.

Esempio

L'esempio 10-4 illustra la convenzione di variabili statiche:

```
VAR
   ESECUZIONE      :INT;
   CAMPO DI MISURA :ARRAY[1..10] OF REAL;
   INTERRUTTORE    :BOOL;
   MOTORE_1,Motore_2:FB100;//Dichiarazione istanza

END_VAR
```

Esempio 10-4 Dichiarazione di variabili statiche

Accesso

L'accesso alle variabili avviene nella parte istruzioni:

- **Accesso dall'interno**: cioè nella parte istruzioni del blocco funzionale nella cui parte dichiarazioni è stata dichiarata la **variabile**. Ciò viene spiegato nel capitolo 14, Assegnazione di valori.
- **Accesso dall'esterno tramite il DB di istanza**: tramite la variabile indicizzata *DBx.variable.* DBx è il nome del blocco dati.

10.6 Variabili temporanee

Panoramica

Le variabili temporanee fanno parte di un blocco di codice e **non** occupano **alcuna** area di memoria statica. Il loro valore rimane inalterato solo per la durata dell'esecuzione del blocco. **Non** è possibile accedere alle variabili temporanee al di fuori del blocco in cui sono state dichiarate le variabili.

Si consiglia di definire dei dati come dati temporanei se essi sono necessari per memorizzare risultati intermedi durante l'elaborazione dei propri OB, FB o FC.

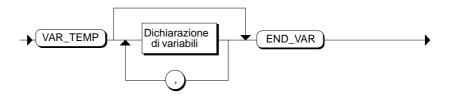
Blocco di variabili temporanee

Non è possibile alcuna inizializzazione

Figura 10-7 Sintassi del blocco di variabili temporanee

Blocco dichiarazioni
VAR_TEMP
END_VAR

Questo blocco dichiarazioni fa parte di un FB, FC, o OB. Conformemente al capitolo 10.2 nell'ambito della dichiarazione di variabili vengono indicati i nomi delle variabili e i tipi di dati (vedere capitolo 10.2).

All'inizio dell'esecuzione di un OB, FB o FC il valore dei dati temporanei non è definito. Non è possibile alcuna inizializzazione.

Esempio

L'esempio 10-5 illustra la dichiarazioni di variabili di blocco temporanee:

```
VAR_TEMP
      BUFFER_1    :ARRAY [1..10] OF INT;
      AIUTO1,AIUTO2:REAL;
END_VAR
```

Esempio 10-5 Dichiarazione di variabili di blocco temporanee

Accesso

L'accesso alle variabili avviene sempre nella parte istruzioni del blocco di codice nella cui parte dichiarazioni è stata dichiarata la **variabile** (**accesso dall'interno**), vedere capitolo 14, Assegnazione di valori.

10.7 Parametri di blocco

Panoramica

I parametri di blocco sono parametri formali di un blocco funzionale o di una funzione. Quando si richiama il blocco funzionale o la funzione, i parametri attuali sostituiscono i parametri formali e costituiscono in tal modo un meccanismo per lo scambio di informazioni fra i blocchi richiamati e i blocchi richamanti.

- I parametri formali d'ingresso assumono i valori d'ingresso attuali (flusso di dati dall'esterno verso l'interno).
- I parametri formali di uscita servono per il trasferimento di valori di uscita (flusso di dati dall'interno verso l'esterno).
- I parametri formali di transito svolgono sia la funzione di parametri d'ingresso sia di parametri di uscita.

Per ulteriori informazioni sull'uso di parametri e sul corrispondente scambio di informazioni si rimanda al capitolo 16.

Figura 10-8 Sintassi di un blocco parametri

Blocco dichiarazioni
`VAR_INPUT`
`VAR_OUTPUT`
`VAR_IN_OUT`

Questo blocco dichiarazioni fa parte di un FB o di una FC nell'ambito della dichiarazione di variabili vengono indicati il nome della variabile e il tipo di dati assegnati (vedere capitolo 10.2).

Dopo la compilazione di un FB, insieme con il blocco `VAR` e `END_VAR` questo blocco determina la struttura del blocco dati di istanza assegnato.

Esempio

L'esempio 10-6 illustra la convenzione di un parametro:

```
VAR_INPUT          //Parametro d'ingresso
  REGOLATORE       :DWORD;
  ORA       :TIME_OF_DAY;
END_VAR

VAR_OUTPUT         //Parametro d'uscita
  VALORI DI RIFERIMENTO: ARRAY [1..10] of INT;
END_VAR

VAR_IN_OUT         //Parametro di transito
  IMPOSTAZIONE: INT;
END_VAR
```

Esempio 10-6 Dichiarazioni di parametri

Accesso

L'accesso ai parametri di blocco avviene nella parte istruzioni di un blocco di codice:

- **Accesso dall'interno**: cioè nella parte istruzioni del blocco nella cui parte dichiarazioni è stato dichiarato il **parametro**. Ciò viene spiegato nel capitolo 14 (Assegnazione di valori) e nel capitolo 13 (Espressioni, operatori e operandi).
- **Accesso dall'esterno**: si può accedere ai parametri di blocchi funzioni tramite i DB di istanza assegnati (vedere capitolo 14.8).

10.8 Flag (flag OK)

Descrizione

Il flag OK serve per prendere nota dell’esecuzione corretta o errata di un blocco. Si tratta di una variabile globale di tipo BOOL con la parola chiave ”OK”.

Se durante l’esecuzione di un’istruzione di blocco si verifica un errore, (p. es. un overflow durante una moltiplicazione) il flag OK viene impostato su FALSE. Quando si esce dal blocco, il valore del flag OK viene memorizzato nel parametro di uscita ENO (capitolo 16.4) definito implicitamente e può quindi essere analizzato dal blocco richiamante.

All’inizio dell’esecuzione di un blocco il flag OK ha il valore TRUE. Tale valore può essere interrogato oppure impostato su TRUE / FALSE in qualsiasi posizione di un blocco con istruzioni SCL.

Convenzione

Il flag OK è una variabile definita dal sistema. Non è necessaria alcuna convenzione. Tuttavia, se si desidera utilizzare il flag OK nel proprio programma utente, prima della compilazione si deve selezionare l’opzione del compilatore ”Attiva Flag OK”.

Esempio

L’esempio 10-7 illustra l’impiego del flag OK:

```
      // Impostare su TRUE il flag OK
      // per poter verificare
      // se l’azione seguente viene svolta
      // correttamente.
OK: = TRUE;
SUM: = SUM + IN;
IF OK THEN
      // L’addizione è stata svolta correttamente
      :
      :
ELSE  // L’addizione non è stata svolta correttamente
      :
END_IF;
```

Esempio 10-7 Impiego del flag OK

11

Definizione di costanti ed etichette di salto

Panoramica

Le costanti sono dati con determinati valori fissi, i quali non possono essere modificati durante l'esecuzione del programma. Se il valore di una costante viene espresso dalla modalità di scrittura, si parla di **costanti literal.**

Non è necessario una dichiarazione di costanti. Sussite, comunque, la possibilità di assegnare dei nomi simbolici alle costanti nella parte dichiarazioni.

Le etichette di salto indicano i nomi di destinazioni di salto all'interno della parte istruzioni del blocco di codice.

I nomi simbolici delle costanti e le etichette di salto vengono definiti, in gruppi separati, in appositi blocchi di dichiarazione.

Sommario del capitolo

11.1 Costanti

Impiego di costanti

Nelle assegnazioni di valori e nelle espressioni, oltre alle variabili e ai parametri di blocco si impiegano anche costanti. Le costanti possono essere impiegate come costanti literal oppure con un nome simbolico.

Definizione di nomi simbolici

La definizione dei nomi simbolici di costanti avviene all'interno del blocco dichiarazioni `CONST` nella parte dichiarazioni del blocco di codice (vedere capitolo 8.4).

Blocco costanti

Figura 11-1 Sintassi di un blocco costanti

La semplice espressione indica, in questo caso, le espressioni matematiche con le quali si possono utilizzare le operazioni fondamentali +, –, *, /, DIV e MOD.

Esempio

L'esempio 11-1 illustra la dichiarazione per i nomi simbolici:

```
CONST
 NUMERO      := 10 ;
 ORA1        := TIME#1D_1H_10M_22S.2MS ;
 NOME        := 'SIEMENS' ;
 NUMERO2     := 2 * 5 + 10 * 4 ;
 NUMERO3     := 3 + NUMERO2 ;
 END_CONST
```

Esempio 11-1 Dichiarazione per costanti simboliche

Modalità di scrittura

SCL dispone di varie modalità di scrittura (formati) per l'introduzione o la visualizzazione di costanti. Queste modalità di scrittura vengono denominate literal. Le seguenti informazioni trattano i singoli literal.

11.2 Literal

Definizione

Un literal rappresenta una modalità di scrittura formale per determinare il valore ed il tipo di costanti. Esistono i seguenti gruppi di literal:

- Literal numerici
- Literal caratteri
- Indicazione di tempo

Per il valore di una costante esiste una determinata modalità di scrittura a seconda del tipo e del formato dei dati:

15	VALORE 15	come numero intero in rappresentazione decimale
2#1111	Valore 15	come numero intero in rappresentazione binaria
16#F	Valore 15	come numero intero in rappresentazione esadecimale

Literal con varie modalità di scrittura per il valore 15

Assegnazione di tipi di dati alle costanti

Ad una costante viene assegnato il tipo di dati il cui campo di valori è appena sufficiente per poter memorizzare la costante senza alcuna perdita di valore. Se si utilizza una costante in una espressione, p. es. in un'assegnazione di valore, il tipo di dati della variabile di destinazione deve accettare il valore della costante. Se, per esempio, viene indicato un literal di numero intero il cui valore eccede il campo dei numeri interi, si presuppone che si tratti di un doppio numero intero. Il compilatore segnalerà un messaggio d'errore se si assegna questo valore ad una variabile di numero intero.

11.3 Modalità di scrittura per literal di numeri interi e literal di numeri in virgola mobile

Panoramica

Per le modalità di scrittura di valori numerici, SCL dispone di:

- literal di numeri interi per valori di numeri interi e
- literal numeri in virgola mobile per numeri in virgola mobile

In entrambi i literal si impiega una sequenza di cifre che deve avere una struttura conforme alla figura 11-2. Nei seguenti diagrammi sintattici, questa sequenza di cifre viene denominata per semplicità **sequenza di cifre decimali**.

Figura 11-2 Sequenza di cifre in un literal

Il punto decimale all'interno di literal si compone di una sequenza di cifre che, come opzione, possono essere separate da trattini. I trattini hanno lo scopo di migliorare la leggibilità di grandi numeri.

Sequenza di cifre decimali

Figura 11-3 Sintassi per sequenze di cifre decimali all'interno di literal

Gli esempi seguenti contengono valide modalità di scrittura per sequenze di cifre decimali all'interno di literal:

```
1000
1_120_200
666_999_400_311
```

Literal di numeri interi

I literal di numeri interi contengono valori di numeri interi. Nel programma SCL, questi ultimi possono essere assegnati (a seconda della lunghezza) a variabili con i tipi di dati:

`BOOL, BYTE, INT, DINT, WORD e DWORD`

La figura 11-4 illustra la sintassi di un literal di numeri interi:

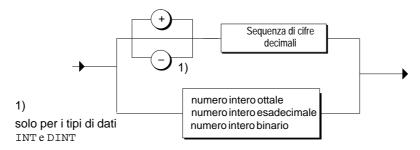



Figura 11-4 Sintassi di un literal di numeri interi

Gli esempi seguenti contengono valide modalità di scrittura per sequenze di cifre decimali all'interno di literal di numeri interi:

```
1000
+1_120_200
–666_999_400_311
```

Numeri binari/ottali/ esadecimali

L'indicazione di un literal di numeri interi in un sistema numerico diverso da quello decimale avviene premettendo i prefissi **2#, 8#** o **16#**, ai quali fa seguito la cifra nella rappresentazione del rispettivo sistema. Il trattino può essere inserito fra le cifre per facilitare la leggibilità di grandi numeri.

La modalità di scrittura generale di un literal di numeri interi viene illustrata nella figura 11-5 con un esempio di una sequenza di cifre per un numero ottale:

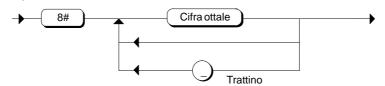



Figura 11-5 Sintassi di una sequenza di cifre ottali

Gli esempi seguenti contengono valide modalità di scrittura per literal di numeri interi:

```
Valore_2:=2#0101;//Numero binario, valore decimale 5
Valore_3:=8#17;  //Numero ottale, valore decimale 14
Valore_4:=16#F;  //Numero esadecimale, valore decimale 15
```

Literal di numeri in virgola mobile

I literal di numeri in virgola mobile sono dei valori con delle cifre dopo la virgola. Essi possono essere assegnati alle variabili con tipo di dati REAL. L'indicazione del segno è opzionale. Se non viene indicato alcun segno, il numero viene interpretato come numero positivo. La figura 11-6 illustra la sintassi per un numero in virgola mobile:

LITERAL DI NUMERI IN VIRGOLA MOBILE

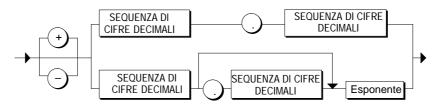



Figura 11-6 Sintassi di un literal di numeri in virgola mobile

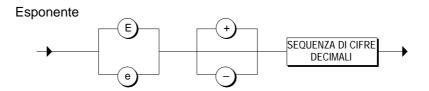
Nella modalità di scrittura esponenziale, per indicare numeri in virgola mobile si può utilizzare un esponente. L'indicazione dell'esponente avviene facendo precedere il numero dalla lettera ”E” o ”e” seguita da un numero decimale. La figura 11-7 illustra la sintassi per l'indicazione dell'esponente.

Esponente

Figura 11-7 Sintassi del esponente

Esempio:

In SCL, il valore 3×10^{10} può essere rappresentato con i seguenti numeri in virgola mobile:

```
3.0E+10     3.0E10     3e+10      3E10
0.3E+11     0.3e11     30.0E+9    30e9
```

Esempi

L'esempio finale 11-2 riepiloga le varie possibilità:

```
//Literal di numeri interi
NUMERO1:= 10 ;
NUMERO2:= 2#1010 ;
NUMERO3:= 16#1A2B ;
//Literal di numeri in virgola mobile
NUMERO4:= -3.4 ;
NUMERO5:= 4e2 ;
NUMERO 6:= 40_123E10;
```

Esempio 11-2 Literal numerici

11.4 Modalità di scrittura per literal di caratteri e di stringhe

Panoramica

SCL ofre anche la possibilità di introdurre ed elaborare informazioni di testo, per esempio una stringa di caratteri che si desidera visualizzare come messaggio.

Con i literal di caratteri non si possono eseguire dei calcoli, il che significa che i literal di caratteri **non** possono essere impiegati in espressioni aritmetiche. Si distingue fra:

- un literal di caratteri, cioè caratteri singoli e
- un literal di stringa, cioè una sequenza di caratteri con max. 254 caratteri singoli.

Literal di caratteri (caratteri singoli)

Il literal di caratteri contiene esattamente un carattere (vedere figura 11-8). Il carattere viene racchiuso fra virgolette semplici (’).

LITERAL DI CARATTERI

Figura 11-8 Sintassi di un literal di caratteri

Esempio:

```
Carattere_1:=’B’;       // Carattere B
```

Literal di stringa

Un literal di stringa è una stringa di caratteri di max. 254 caratteri (lettere, cifre e caratteri speciali), racchiusi fra virgolette (’). Si possono utilizzare sia caratteri minuscoli che maiuscoli.

LITERAL DI STRINGA

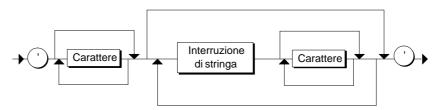



Figura 11-9 Sintassi di un literal di stringa

Seguono alcuni esempi di literal di stringa validi :

```
’ROT’       ’76181 Karlsruhe’       ’270-32-3456’
’DM19.95’   ’La risposta giusta è:’
```

Si deve tener presente che in un’assegnazione di un literal di stringa ad una variabile di stringa, il numero dei caratteri può essere limitato a < 254. L’assegnazione di valore:

TESTO:STRINGA[20]:='SIEMENS KARLSRUHERheinbrückenstr.'

causa un messaggio di errore e modifica la variabile 'TEXT' in

'SIEMENS KARLSRUHE Rh'

I caratteri di formattazione speciali, le virgolette (') o un carattere $ possono essere introdotti con il segno del dollaro $. Un literal di stringa può essere più volte interrotto e continuato.

Interruzione di stringa

Una stringa è situata in una riga di un blocco SCL oppure viene ripartita su più righe mediante identificazioni speciali. Per interrompere una stringa viene usata l'identificazione '$>', per continuare una stringa si usa l'identificazione '$<'.

TESTO:STRINGA[20]:='L'FB$>//Versione preliminare

$<convertito';

Lo spazio fra l'identificazione di interruzione e l'identificazione di continuazione può estendersi anche su più righe e oltre a spazi vuoti può contenere solo commenti. In tal modo, si può interrompere e continuare più volte un literal di stringa (vedere anche la figura 11-10).

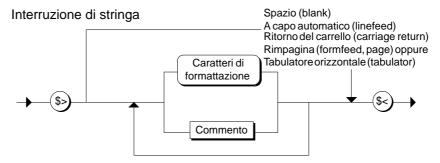



Figura 11-10 Sintassi dell'interruzione di stringa

Caratteri stampabili

I caratteri di un literal di caratteri o di stringa possono essere dei caratteri qualsiasi contenuti nel set di caratteri ASCII completo. I caratteri di formattazione speciali e i caratteri non rappresentabili direttamente (' e $) devono essere introdotti con l'ausilio del segno del dollaro $.

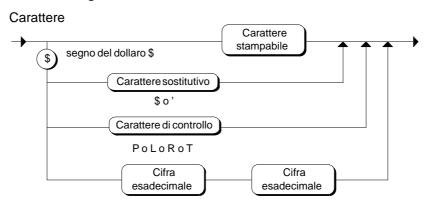



Figura 11-11 Sintassi di un carattere

Caratteri non stampabili

In un literal di caratteri si possono introdurre anche caratteri non stampabili del set di caratteri ASCII ampliato. A tal fine si deve utilizzare la rappresentazione sostitutiva in codice esadecimale.

Un carattere ASCII viene introdotto con i caratteri $hh, in cui **hh** indica il valore esadecimale del carattere ASCII.

Esempio:

```
CARATTERI_A :='$41'; //corrisponde al carattere 'A'
Blank       :='$20';.//corrisponde al carattere ␣
```

Per ulteriori informazioni sui caratteri sostitutivi e di controllo si rimanda all'appendice A.

Esempi

Gli esempi seguenti illustrano la formulazione dei literal di caratteri:

```
// Literal di caratteri
Caratteri:= 'S' ;

// Literal di stringa:
NOME:= 'SIEMENS' ;

// Interruzione di un literal di stringa:
MESSAGGIO1:= 'MOTORE- $>
$< Controllo' ;

// Stringa in rappresentazione esadecimale:
MESSAGGIO1:= '$41$4E' (*Sequenza di caratteri AN*);
```

Esempio 11-3 Literal di caratteri

11.5 Modalità di scrittura per indicazioni del tempo

Modalità di scrittura per valori di tempo

SCL dispone di vari formati per introdurre i valori dell'ora e della data. Sono possibili le seguenti indicazioni di tempo:

```
Data
Durata
Ora del giorno
Data e ora
```

Data

Una data viene preannunciata con i prefissi DATE# o D# (vedere la figura 11-12).

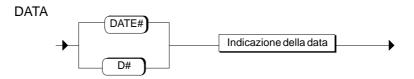



Figura 11-12 Sintassi per la data

L'**indicazione della data** avviene tramite numeri interi per la cifra dell'anno (di 4 cifre), indicazione della data e del giorno, i quali devono essere separati mediante trattini.

Figura 11-13 Sintassi per l'indicazione della data

Delle indicazioni valide per la data potrebbero essere:

```
// Indicazione della data
VARIABILE DI TEMPO1:= DATE#1995-11-11;
VARIABILE DI TEMPO2:= D#1995-05-05;
```

Durata

La durata, conformemente alla figura 11-14 viene introdotta dal prefisso TIME# o T#. L'indicazione della durata può aver luogo in due modi:

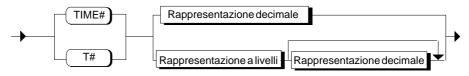
- rappresentazione decimale
- rappresentazione a livelli

- Ogni unità di tempo (p. es. ore, minuti) può essere indicata una sola volta.
- La sequenza - giorni, ore, minuti, secondi, millisecondi - deve essere rispettata

Figura 11-14 Sintassi per la durata

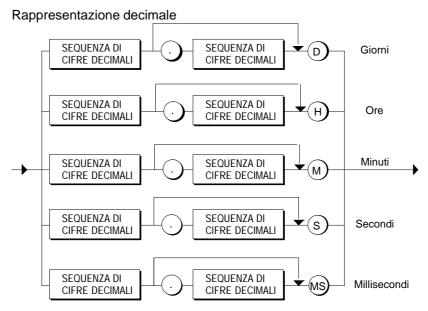
La **rappresentazione decimale** viene utilizzata quando si deve indicare la durata con giorni, ore, minuti, secondi o millisecondi:

L'accesso alla rappresentazione decimale è possibile solo con unità di tempo non ancora definite.

Figura 11-15 Sintassi per la rappresentazione decimale

Esempi

Alcune indicazioni valide potrebbere essere:

```
TIME#20.5D          per 20,5  giorni
TIME#45.12M         per 45,12 minuti
T#300MS             per 300 millisecondi
```

La **rappresentazione a livelli** viene impiegata quando si deve indicare la durata come una sequenza di giorni, ore, minuti, secondi o millisecondi (vedere figura 11-16). L'omissione di singoli componenti è consentita. Si deve però indicare almeno una componente.

Rappresentazione a livelli

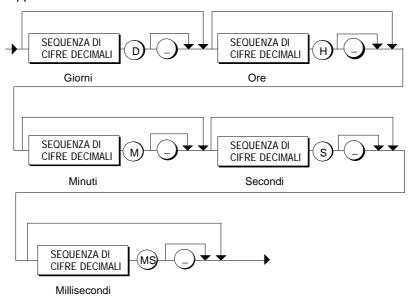



Figura 11-16 Sintassi per la rappresentazione a livelli

Seguono alcune valide indicazioni:

```
TIME#20D_12H
TIME#20D_10H_25M_10s
TIME#200S_20MS
```

Ora del giorno

L'indicazione dell'ora del giorno viene preannunciata dai prefissi TIME_OF_DAY# o TOD# (vedere figura 11-17).

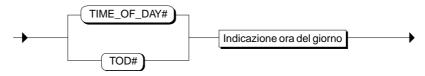



Figura 11-17 Sintassi per l'ora del giorno

L'**indicazione dell'ora** avviene introducendo ore, minuti e secondi, i quali devono essere separati mediante doppio punto. L'indicazione dei millisecondi è opzionale. Essa viene separata dagli altri componenti tramite un punto. La figura 11-18 illustra la sintassi per l'indicazione dell'ora del giorno:

Figura 11-18 Sintassi per l'indicazione dell'ora

Alcune indicazioni valide potrebbero essere:

```
//Indicazione ora del giorno
ORA1:= TIME_OF_DAY#12:12:12.2;
ORA2:= TOD#11:11:11.7.200;
```

Data e ora

L'indicazione della data e dell'ora viene preannunciata dai prefissi DATE_AND_TIME# o DT# (vedere figura 11-19). Si tratta di un literal composto dalle indicazioni di data e ora.

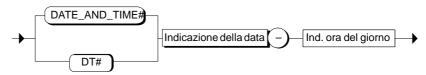



Figura 11-19 Sintassi per la data e l'ora

L'esempio illustra l'indicazione di data e ora:

```
// Indicazione dell'ora
ORA1:= DATE_AND_TIME#1995-01-01-12:12:12.2;
ORA2:= DT#1995-02-02-11:11:11;
```

11.6 Etichette di salto

Descrizione

Le etichette di salto (Labels) servono per definire la destinazione di una istruzione GOTO (vedere capitolo 11-4).

Definizione di etichette di salto

Le etichette di salto vengono definite con il loro nome simbolico nella parte dichiarazioni del blocco di codice (vedere capitolo 8.4):

Blocco etichette di salto

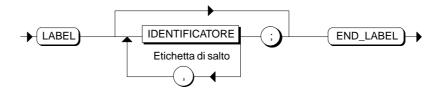



Figura 11-20 Sintassi per blocco etichette di salto

Esempio

L'esempio seguente illustra la convenzione per le etichette di salto:

```
LABEL
ETICHETTA1, ETICHETTA2, ETICHETTA3;
END_LABEL;
```

Esempio 11-4 Etichette di salto

12 Definizione di dati globali

Panoramica

I dati globali sono dati che possono essere utilizzati da qualsiasi blocco di codice (FC, FB, OB). L'accesso a questi dati può aver luogo in modo assoluto o simbolico. In questo capitolo vengono descritte le singole aree di dati. Viene inoltre descritto il modo di accesso a tali dati.

Sommario del capitolo

12.1 Panoramica

Dati globali

In SCL si ha la possibilità di accedere ai dati globali. Esistono due tipi di dati globali:

- **Aree di memoria della CPU**

  Queste aree di memoria sono dati definiti dal sistema: p. es. ingressi, uscite e merker (vedere capitolo 7.5). La dimensione delle aree di memoria a disposizione dell'utente dipende dalla CPU impiegata.

- **Dati utente globali come blocchi dati caricabili**

  Queste aree di dati sono situate all'interno di blocchi dati. Per poterle utilizzare occorre dapprima creare i blocchi dati e definire i propri dati in tali blocchi. Nel caso di blocchi dati di istanza, essi vengono derivati da blocchi funzionali e generati automaticamente.

Tipi di accesso

L'accesso ai dati globali può aver luogo nei modi seguenti:

- **assoluto:** mediante identificazione di operando e indirizzo assoluto.
- **simbolico:** mediante un simbolo che l'utente ha definito in precedenza nella tabella dei simboli (vedere **/231/**).
- **indicizzato:** mediante identificazione di operando e indice di campo.
- **strutturato:** mediante una variabile.

La tabella seguente illustra l'impiego dei vari tipi di accesso:

Tabella 12-1 Impiego dei tipi di accesso ai dati globali

Tipo di accesso	Aree di memoria della CPU	Dati utente globali
assoluto	si	si
simbolico	si	no
indicizzato	si	si
strutturato	no	si

12.2 Aree di memoria della CPU

Definizione

Le aree di memoria della CPU sono aree definite dal sistema. Perciò, l'utente non deve definire tali aree nel proprio blocco di codice.

Aree di memoria della CPU

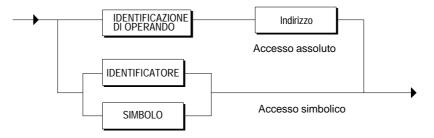
Ogni CPU mette a disposizione le seguenti aree di memoria con una propria area di indirizzamento:

- Ingressi / uscite nell'immagine di processo
- Ingressi / uscite di periferia
- Merker
- Temporizzatori, contatori (vedere capitolo 17)

Sintassi per l'accesso

L'accesso ad un'area di memoria della CPU avviene mediante un'assegnazione di valori nella parte istruzioni di un blocco di codice (vedere capitolo 14.3):

- con un semplice accesso, che può essere del tipo assoluto o simbolico, oppure
- con un accesso indicizzato.

SEMPLICE ACCESSO ALLA MEMORIA

ACCESSO INDICIZZATO ALLA MEMORIA

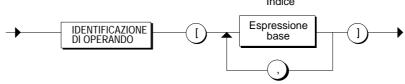



Figura 12-1 Sintassi per i tipi di accesso alle aree di memoria della CPU

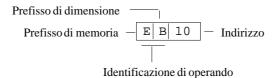
12.3 Accesso assoluto alle aree di memoria della CPU

Principio

L'accesso assoluto alle aree di memoria della CPU avviene mediante assegnazione di valore di un identificatore assoluto ad una variabile dello stesso tipo:

L'identificatore assoluto fa riferimento ad un'area di memoria all'interno della CPU. L'utente può specificare quest'area indicando l'identificazione di operando (qui `EB`) seguita dall'indirizzo (qui `10`).

Identificatore assoluto

L'identificatore assoluto si compone di un'identificazione di operando con prefissi di memoria e di dimensione nonché di un indirizzo.

Identificazione di operando

La combinazione di prefisso di memoria e prefisso di dimensione forma l'identificazione di operando.

Identificazione di operando

Figura 12-2 Sintassi di un'identificazione di operando

Prefisso di memoria

Tramite il prefisso di memoria si definisce il tipo dell'area di memoria da indirizzare. Conformemente alla figura 12-3 sono disponibili i seguenti tipi:**1)**

Prefisso di memoria

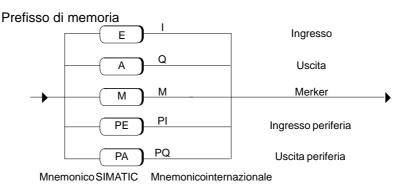



Figura 12-3 Sintassi di un prefisso di memoria

1) *Indipendentemente dall'impostazione della lingua nel SIMATIC Manager, le identificazioni di operando SIMATIC o IEC hanno un significato riservato. E' possibile, infatti, importare in SIMATIC Manager, la lingua ed il mnemonico separatamente.*

Prefisso di dimensione

Mediante indicazione del prefisso di dimensione si specifica la dimensione risp., il tipo dell'area di memoria che deve essere letta dalla periferia, p. es. un byte o una parola. L'indicazione del prefisso di dimensione è opzionale se si specifica un bit. La figura 12-4 illustra la sintassi:

Prefisso di dimensione

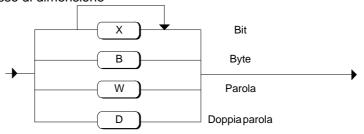



Figura 12-4 Sintassi di un prefisso di dimensione

Indirizzo

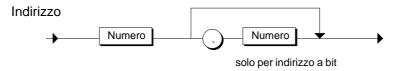
Per l'indicazione dell'indirizzo si deve indicare, in funzione del prefisso di dimensione utilizzato, un indirizzo assoluto indicante un bit, byte, una parola o una doppia parola. Solo se è stato specificato "Bit" si può indicare un indirizzo a bit supplementare (vedere figura 12-5). Il primo numero significa indirizzo di byte ed il secondo indirizzo a bit.

Indirizzo

Figura 12-5 Sintassi di un indirizzo

Esempi

Seguono alcuni esempi di accesso assoluto:

```
STATUSBYTE  := EB10;
STATUS_3    := E1.1;
Valoremisura      := EW20;
```

Esempio 12-1 Accesso assoluto

12.4 Accesso simbolico alle aree di memoria della CPU

Principio

L'indirizzamento simbolico consente di utilizzare dei nomi simbolici al posto di identificatori assoluti per indirizzare le aree di memoria della CPU:

Simbolo	Indirizzo assoluto	Tipo di dati	Commento
Contatto motore	E 1.7	BOOL	Interruttore a contatto 1 per motore A
Ingresso	EW 10	INT	Parola di stato d'ingresso
Byte di ingresso 1	EB 1	BYTE	Byte di ingresso
”Ingresso 1.1”	E 1.1	BOOL	Barriera luminosa
Canali di misura	MW 2	REAL	Buffer

L'assegnazione dei nomi simbolici ai singoli operandi del programma utente avviene mediante creazione di un'apposita tabella dei simboli.

Come tipi di dati sono consentiti tutti i tipi di dati semplici purché essi siano in grado di recepire la larghezza di accesso.

Accesso

L'accesso avviene p. es. mediante assegnazione di valore ad una variabile dello stesso tipo con simbolico predefinito.

```
VALOREDIMISURA_1        := Contatto motore 1;
```

Creazione della tabella dei simboli

La creazione della tabella dei simboli e l'introduzione dei valori nella tabella dei simboli avviene con STEP 7.

La tabella dei simboli può essere attivata con il SIMATIC Manager o direttamente con SCL tramite il comando di menu **Strumenti ▸ Tabella dei simboli**

Inoltre, è possibile importare e sottoporre ad ulteriore elaborazione la tabella dei simboli presente come file di testo che puó essere creata con qualsiasi editor di testi (per informazioni al riguardo vedere **/231/**).

Esempi

Seguono alcuni esempi di accesso simbolico:

```
STATUSBYTE  := Byte d'ingresso;
STATUS_3    := ”Ingresso 1.1”;
Messwert    := Canali di misura;
```

Esempio 12-2 Accesso simbolico

12.5 Accesso indicizzato alle aree di memoria della CPU

Principio

Esiste anche la possibilità di accedere in modo indicizzato a specifiche aree della CPU. Rispetto all'indirizzamento assoluto, questo metodo offre il vantaggio di poter indirizzare dinamicamente gli indirizzi mediante impiego di indici variabili. Per es. come indirizzo si può utilizzare la variabile di esecuzione di un loop FOR.

L'accesso indicizzato ad un'area di memoria è simile all'accesso assoluto. L'unica differenza consiste nell'indicazione dell'indirizzo. Al posto dell'indirizzo viene specificato un indice che può essere una costante, una variabile o un'espressione aritmetica.

Identificatore assoluto

Nell'accesso indicizzato, l'identificatore assoluto si compone dell'identificazione di operando e di un'espressione base per l'indicizzazione (secondo il capitolo 12.3).

Regole per l'accesso indicizzato

L'indicizzazione deve essere conforme alle regole seguenti:

- Per un accesso con tipo di dati `BYTE`, `WORD` o `DWORD` si deve utilizzare esattamente un indice. L'indice viene interpretato come indirizzo a byte. La larghezza di accesso viene definita tramite il prefisso di dimensione.
- In caso di accesso con tipo di dati `BOOL` si devono utilizzare due indici. Il primo indice specifica l'indirizzo a byte, mentre il secondo indice specifica la posizione del bit all'interno del byte.
- Ogni indice deve essere un'espressione aritmetica del tipo di dati `INT`.

```
PAROLAMISURA_1    := EW[CONTATORE];

MARCHIO     := E[NBYTE, NBIT];
```

Esempio 12-3 Accesso indicizzato

12.6 Blocchi dati

Panoramica

Nei **blocchi dati**, a seconda del caso specifico, si possono memorizzare ed elaborare tutti i dati il cui campo di validità fa riferimento all'intero programma o all'intero progetto. Ogni blocco di codice può accedere in lettura o scrittura ai dati utente globali.

Convenzione

Per la sintassi concernente la struttura di blocchi dati si rimanda al capitolo 8. Si deve distinguere fra due diversi tipi di blocchi dati:

- blocchi dati
- blocchi dati di istanza

Accesso ai blocchi dati

L'accesso ai dati di un qualsiasi blocco dati avviene sempre nel modo seguente:

- semplice o assoluto
- indicizzato
- strutturato

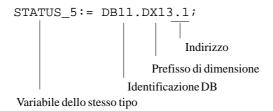
La figura 12-6 contiene una panoramica dei vari tipi di accesso:

Sintassi

Figura 12-6 Sintassi dei tipi di accesso ai blocchi dati

12.7 Accesso assoluto ai blocchi dati

Principio

L'accesso assoluto a un blocco dati avviene in modo analogo all'accesso alle aree di memoria della CPU mediante assegnazione di un valore ad una variabile dello stesso tipo. Dopo l'introduzione dell'identificazione DB segue la parola chiave ″D″ con indicazione del prefisso di dimensione (p. es. X per BIT) e l'indirizzo a byte (p. es. 13.1).

Accesso

L'accesso viene specificato indicando l'identificazione DB insieme con il prefisso di dimensione e l'indirizzo (vedere figura 12-7).

Accesso assoluto al DB

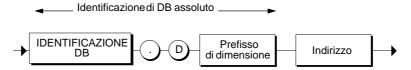



Figura 12-7 Sintassi di un accesso assoluto al DB

Prefisso di dimensione

Indica la dimensione dell'area di memoria nel blocco dati che deve essere indirizzato, p. es. un byte o una parola. L'indicazione del prefisso di dimensione è opzionale se si desidera specificare un bit. La figura 12-8 illustra la sintassi per il prefisso di dimensione.

Prefisso di dimensione

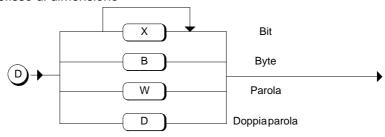



Figura 12-8 Sintassi del prefisso di dimensione per un blocco dati

Indirizzo

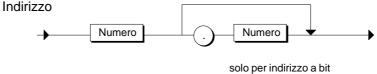
Per l'indicazione dell'indirizzo conformemente alla figura 12-9, in funzione del prefisso di dimensione utilizzato, si deve introdurre un indirizzo assoluto che fa riferimento a un bit, a un byte, a una parola o a una doppia parola. Solo se è stato specificato "Bit" si può introdurre un indirizzo a bit supplementare. Il primo numero indica l'indirizzo a byte mentre il secondo indica l'indirizzo a bit.

Indirizzo

Figura 12-9 Sintassi di un indirizzo

Esempi

Seguono alcuni esempi di accesso assoluto ad un blocco dati. Nella prima parte il blocco dati viene indicato in modo assoluto, nella seconda parte in modo simbolico:

```
STATUSBYTE        := DB101.DB10;
STATUS_3          := DB30.D1.1;
Valoremisura      := DB25.DW20;

STATUSBYTE        := Datidistato.DB10;
STATUS_3          := ”Nuovi dati” D1.1;
Valore di misura  := Dati di misura.DW20;

STATUS_1          := WORD_TO_BLOCK_DB (INDEX).DW10;
```

Esempio 12-4 Accesso assoluto

12.8 Accesso indicizzato ai blocchi dati

Accesso indicizzato

Esiste anche la possibilità di accedere a DB globali in modo **indicizzato**. Rispetto all'indirizzamento assoluto, questo metodo offre il vantaggio di poter indirizzare dinamicamente lo scambio di dati mediante l'impiego di indici variabili. Per es. come indirizzo si può utilizzare la variabile di esecuzione di un loop FOR.

L'accesso indicizzato ad un blocco dati avviene in modo analogo all'accesso assoluto. L'unica differenza consiste nell'indicazione dell'indirizzo.

Al posto dell'indirizzo viene specificato un indice che può essere una costante, una variabile o un'espressione aritmetica.

Identificatore assoluto

Nell'accesso indicizzato, l'identificatore assoluto si compone dell'identificazione di operando (secondo il capitolo 12.7) e di un'espressione base per l'indicizzazione.

Regole per l'accesso indicizzato

L'indicizzazione deve essere conforme alle regole seguenti:

- Ogni indice deve essere un'espressione aritmetica del tipo di dati `INT`.
- Per un accesso con tipo di dati `BYTE`, `WORD` o `DWORD`, si deve utilizzare esattamente un indice. L'indice viene interpretato come indirizzo a byte. La larghezza di accesso viene definita tramite il prefisso di dimensione.
- In caso di accesso con tipo di dati `BOOL` si devono utilizzare due indici. Il primo indice specifica l'indirizzo a byte, mentre il secondo indice specifica la posizione del bit all'interno del byte.

```
STATUS_1:= DB11.DW[CONTATORE];
STATUS_2:= DB12.DW[WNR, NBIT];

STATUS_1:= Base dati1.DW[CONTATORE];
STATUS_2:= Base dati2.DW[WNR, NBIT];

STATUS_1:= WORD_TO_BLOCK_DB (INDEX).DW[CONTATORE];
```

Esempio 12-5 Accesso indicizzato

12.9 Accesso strutturato ai blocchi dati

Principio

L'accesso strutturato avviene mediante un'assegnazione di valori ad una variabile dello stesso tipo.

```
TEMPO_1:= DB11.ORA ;
```


Per far riferimento ad una variabile nel blocco dati è sufficiente indicare il nome del DB e, separato da un punto, il nome della variabile semplice. La relativa sintassi viene descritta nella figura 12-6.

Per variabile semplice qui si intende una variabile alla quale nella dichiarazione DB è stato assegnato un tipo di dati semplici o composti.

Esempi

```
Nella parte dichiarazioni dell'FB10:
VAR
Risultato:  STRUCT ERG1 : INT;
                   ERG2 : WORD;
                   END_STRUCT
 END_VAR
```

```
Tipo di dati definito dall'utente UDT1:
TYPE UDT1   STRUCT ERG1 : INT;
                   ERG2 : WORD;
                   END_STRUCT
```

```
DB20 con tipo di dati definito dall'utente:
DB20
UDT1
BEGIN ...
```

```
DB30 senza tipo di dati definito dall'utente:
DB30        STRUCT ERG1 : INT;
                   ERG2 : WORD;
                   END_STRUCT
BEGIN ...
```

Esempio 12-6 Dichiarazione per i dati dei blocchi dati

```
Blocco funzionale con gli accessi:
..
FB10.DB10();
PAROLARIS_A :=    DB10.Risultato.ERG2;
PAROLARIS_B :=    DB20.ERG2;
PAROLARIS_C :=    DB30.ERG2;
```

Esempio 12-7 Accesso ai dati dei blocchi dati

Espressioni, operatori e operandi 13

Panoramica

Una espressione indica un valore che viene calcolato al momento della compilazione o dell'esecuzione del programma. Esso si compone di operandi (p. es. costanti, variabili o valori di funzioni) e operatori (p. es. *, /, +, –).

I tipi di dati degli operandi e degli operatori interessati determinano il tipo di espressione. SCL distingue tra:

- espressioni aritmetiche
- espressioni di potenza
- espressioni di confronto
- espressioni logiche

Sommario del capitolo

13.1 Operatori

Panoramica

Le espressioni si compongono di operatori e operandi. La maggior parte degli operatori di SCL sono una combinazione di due operandi e vengono perciò denominati *binari*. Gli altri operatori lavorano con un solo operando e vengono perciò denominati *unari*.

Gli operatori binari vengono scritti fra gli operandi (p. es. A + B). Un operatore unario è sempre situato prima del suo operando (p. es. –B).

La priorità degli operatori descritta nella tabella 13-1 regola la sequenza dei calcoli. La cifra ”1” corrisponde alla massima priorità.

Tabella 13-1 Panoramica degli operatori

Classi di operatori

Classe	Operatore	Rappresentazione	Priorità
Operatore di assegnazione: *Questo operatore assegna un valore ad una variabile*	Assegnazione	: =	11
Operatori aritmetici	Potenza	**	2
	Operatori unari		
	Più unario	+	3
Questi operatori servono per calcoli matematici	Meno unario	-	3
	Operatori aritmetici di base		
	Moltiplicazione	*	4
	Funzione modulo	MOD	4
	Divisione numeri interi	DIV	4
	Addizione	+	5
	Sottrazione	-	5
Operatori di confronto *Questi operatori sono necessari per poter formulare delle condizioni*	Minore di Maggiore di Minore o uguale Maggiore o uguale Uguaglianza Disuguaglianza	< > <= >= = <>	6 6 6 6 7 7
Operatori logici	Negazione (unario)	NOT	3
	Operatori logici di base		
Questi operatori sono necessari per espressioni logiche	AND OR esclusivo OR	AND o & XOR OR	8 9 10
Caratteri di parentesi	(**Espressione**)	( )	1

13.2 Sintassi delle espressione

Panoramica

Le espressioni possono essere rappresentate con il diagramma sintattico illustrato nella figura 13-1. Le espressioni aritmetiche, logiche e di confronto e le espressioni di potenza presentano alcuni aspetti particolari e perciò vengono descritte separatamente nei capitoli da 13.3 fino a 13.6.

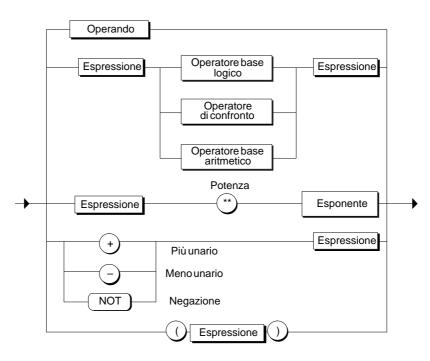



Figura 13-1 Sintassi di espressioni

Risultato di un'espressione

Il risultato di un'espressione può essere:

- assegnato ad una variabile
- utilizzato come condizione per una istruzione di controllo
- utilizzato come parametro per il richiamo di una funzione o di un blocco funzionale.

Sequenza di analisi

La sequenza di analisi di un'espressione dipende:

- dalla priorità degli operatori interessati e
- dalla sequenza sinistra-destra o
- dalla parentesi usata (per operatori della stessa priorità).

Regole

L'elaborazione delle espressioni avviene conformemente alle seguenti regole:

- Gli operandi vengono elaborati conformemente alla loro priorità.
- Gli operatori della stessa priorità vengono elaborati da sinistra verso destra.
- L'inserimento del segno meno davanti a un identificatore equivale ad una moltiplicazione per –1.
- Gli operatori aritmetici non devono essere disposti in successione. Perciò, l'espressione a * – b non è valida, mentre invece a*(–b) è consentita.
- L'inserimento di coppie di parentesi può disattivare la priorità degli operatori, cioè le parentesi hanno la massima priorità.
- Le espressioni fra parentesi vengono considerate come operandi singoli e vengono analizzate sempre per prime.
- Il numero di parentesi sinistre deve corrispondere al numero di parentesi destre.
- Gli operatori aritmetici non possono essere applicati a caratteri o a dati logici. Perciò, le espressioni come 'A' + 'B' e (n<=0) + (m<0) sono errate.

Esempi

Le varie espressioni possono avere la seguente struttura:

```
EB10                       // Operando
A1 AND (A2)                // Espressione logica
(A3) < (A4)                // Espressione di confronto
3+3*4/2                    // Espressione aritmetica

VALOREMISURA**2            // Espressione di potenza
(DIFFERENZA)**DB10.ESPONENTE // Espressione di potenza
(SOMMA)**FC100(..)         // Espressione di potenza
```

Esempio 13-1 Varie espressioni

13.2.1 Operandi

Panoramica

Gli operandi sono oggetti con i quali si possono creare espressioni. Gli operandi possono essere rappresentati con un diagramma sintattico (vedere figura 13-2).

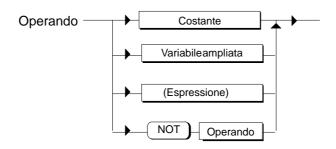



Figura 13-2 Sintassi di operandi

Costante

Si possono utilizzare costanti con il loro valore numerico, con un nome simbolico o con una sequenza di caratteri.

Figura 13-3 Costanti come operandi

Seguono alcuni esempi di costanti valide:

```
4_711
4711
30.0
'CARATTERE'
FATTORE
```

Variabile ampliata

La variabile ampliata è un termine generico per indicare una serie di variabili che vengono descritte più dettagliatamente nel capitolo 14.

Variabile ampliata

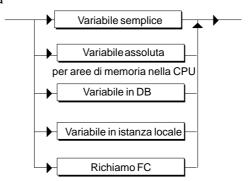



Figura 13-4 Sintassi per la variabile ampliata

Esempi di variabili ampliate

Seguono alcuni esempi di variabili valide:

```
VALORERIFERIMENTO        variabile semplice
EW10                     Variabile assoluta
E100.5                   Variabile assoluta
DB100.DW[INDICE]         Variabile nel DB
NGIRI.MOTORE             Variabile nell'istanza locale
SQR(20)                  Funzione standard
FC192(VALORERIFERIMENTO) Richiamo della funzione
```

Esempio 13-2 Variabile ampliata in espressioni

Avvertenza

In un richiamo della funzione, il risultato calcolato, ossia il valore di ritorno, viene inserito nell'espressione al posto del nome della funzione. Per tale motivo, le funzioni VOID che non possiedono alcun valore di ritorno **non** sono ammesse come operandi di un'espressione.

13.3 Espressioni aritmetiche

Definizione

Un'espressione aritmetica è un'espressione formata con operatori aritmetici. Queste espressioni consentono l'elaborazione di tipi di dati numerici.

Operatore aritmetico di base

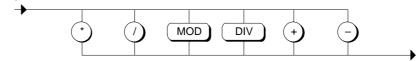



Figura 13-5 Sintassi di un operatore aritmetico di base

Operazioni aritmetiche

La tabella 13-2 contiene un riepilogo delle operazioni possibili e mostra a quale tipo si deve assegnare il risultato in funzione degli operandi. Vengono usate le abbreviazioni seguenti:

`ANY_INT` per i tipi di dati `INT, DINT`

`ANY_NUM` per i tipi di dati `ANY_INT e REAL`

Tabella 13-2 Operatori aritmetici

Operazione	Operatore	1º Operando	2º Operando	Risultato [1]	Priorità
Potenza	**	ANY_NUM	INT	REAL	2
Più unario	+	ANY_NUM TIME	- -	ANY_NUM TIME	3
Meno unario	-	ANY_NUM TIME	- -	ANY_NUM TIME	3
Moltiplicazione	*	ANY_NUM	ANY_NUM	ANY_NUM	4
		TIME	ANY_INT	TIME	
Divisione	/	ANY_NUM	ANY_NUM	ANY_NUM	4
Divisione di interi	DIV	ANY_INT	ANY_INT	ANY_INT	4
		TIME	ANY_INT	TIME	
Divisione modulo	MOD	ANY_INT	ANY_INT	ANY_INT	4
Addizione	+	ANY_NUM	ANY_NUM	ANY_NUM	5
		TIME	TIME	TIME	
		TOD	TIME	TOD	
		DT	TIME	DT	
Sottrazione	–	ANY_NUM	ANY_NUM	ANY_NUM	5
		TIME	TIME	TIME	
		TOD	TIME	TOD	
		DATE	DATE	TIME	
		TOD	TOD	TIME	
		DT	TIME	DT	
		DT	DT	TIME	

[1] *Si deve tener presente che il risultato viene determinato dal più importante tipo di operando.*

Regole

Gli operatori delle espressioni aritmetiche vengono trattati in base alla loro priorità (vedere tabella 13-2).

- Per una maggiore chiarezza, si raccomanda di racchiudere fra parentesi i numeri negativi anche nei casi in cui ciò non è necessario.
- Nelle divisioni con due operandi interi del tipo di dati INT, gli operatori ”DIV” e ”/” forniscono lo stesso risultato (vedere esempio 13-3).
- Gli operatori di divisione (”/”, MOD e ”DIV”) richiedono che il secondo operando sia diverso da zero.
- Se un operando è del tipo INT (numero intero) e l’altro operando è del tipo REAL (numero in virgola mobile), il risultato sarà sempre di tipo REAL.

Esempi

Gli esempi seguenti illustrano la formazione di espressione aritmetiche.

Supponiamo che i e j siano variabili di numeri interi con valori di 11 o –3. Nell’esempio 13-3 riportato vengono illustrate alcune espressioni di numeri interi e i loro valori corrispondenti.

```
Espressione     Valore

 i + j             8
 i – j            14
 i * j           –33
 i DIV j          –3
 i MOD j           2
 i/j              –3
```

Esempio 13-3 Espressioniaritmetiche

Supponiamo che i e j siano variabili di numeri interi con valori di 3 o –5. In tal caso, il risultato dell’espressione aritmetica in base all’esempio 13-4, cioè il valore intero 7, viene assegnato alla variabile VALORE.

```
VALORE:= i + i * 4 / 2 - (7+i) / (-j) ;
```

Esempio 13-4 Espressioneartimetica

13.4 Esponenti

Panoramica

La figura 13-6 ha il compito di illustrare la formazione dell'esponente in un'espressione di potenza (vedi capitolo 13.2). Si deve tener presente che l'esponente può essere formato anche con variabili ampliate.

Esponente

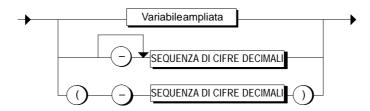



Figura 13-6 Sintassi: esponente

```
 VALOREMISURA**2              // Espressione di potenza
(DIFFERENZA)**DB10.ESPONENTE // Espressione di potenza
(SOMMA)**FC100               // Espressione di potenza
```

Esempio 13-5 Espressioni di potenza con vari esponenti

13.5 Espressioni di confronto

Definizione

Un'espressione di confronto è un'espressione del tipo BOOL generata con operatori di confronto. Queste espressioni vengono generate mediante combinazione di operandi dello stesso tipo con gli operatori elencati nella figura 13-7.

Operatore di confronto

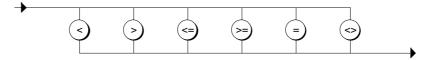



Figura 13-7 Sintassi di un operatore di confronto

Operazioni di confronto

Gli operatori di confronto confrontano gli operandi con riferimento al loro valore numerico.

`1.operando operatore 2.operando ⇒ valore booleano`

Come risultato si ottiene un valore che può essere vero o falso. Il risultato è vero (TRUE) se il confronto è positivo e falso (FALSE) se il confronto è negativo.

Regole

Per la generazione di espressioni di confronto si devono rispettare le seguenti regole:

- Per assicurare la massima chiarezza nella sequenza in cui vengono eseguite le operazioni logiche, gli operandi logici dovrebbero essere racchiusi fra parentesi.
- Le espressioni logiche possono essere combinate in base alla legge della logica booleana per poter realizzare interrogazioni del tipo "se a < b **e** b < c, allora ...". Come espressioni si possono utilizzare variabili o costanti di tipo BOOL e espressioni di confronto.
- I confronti sono consentiti solo fra espressioni della stessa classe, ossia:
  - `INT, DINT, REAL`
  - `BOOL, BYTE, WORD, DWORD`
  - `CHAR, STRING`
- Per questo tipo di variabili, i confronti sono consentiti solo fra variabili dello stesso tipo:
  - `DATE, TIME, TOD, DT`
- Nei confronti di caratteri (tipo CHAR) l'analisi viene eseguita in base alla sequenza di caratteri ASCII.
- Le variabili S5TIME non sono confrontabili.
- Se entrambi gli operandi sono del tipo `DT` o `STRING`, il confronto deve essere eseguito con le funzioni IEC corrispondenti.

Esempi

Gli esempi seguenti illustrano la formazione delle espressioni di confronto:

```
// Il risultato dell’espressione di confronto
// viene negato.

      IF NOT (CONTATORE > 5) THEN... ;



// Il risultato della prima espressione di
// confronto viene negato e congiunto
// con il risultato della seconda espressione

      A:= NOT (CONTATORE1 = 4) AND (CONTATORE2 = 10) ;



// Disgiunzione di due espressioni di confronto
      WHILE (A >= 9) OR (INTERROGAZIONE <> ’n’) DO... ;
```

Esempio 13-6 Espressioni logiche

13.6 Espressioni logiche

Definizione

Un'espressione logica è un'espressione generata con operatori logici. Con gli operatori (AND, &, XOR e OR) si possono combinare operandi logici (tipo BOOL) o variabili del tipo di dati BYTE, WORD o DWORD, per poter generare espressioni logiche. Per negare il valore di un operando logico, cioè per invertirlo, viene utilizzato l'operatore NOT.

Operatore logico di base

NOT non è un operatore di base!
L'operatore funge da segno matematico.

AND | & | XOR | OR

Figura 13-8 Sintassi di un operatore logico di base

Operazioni logiche

La tabella 13-3 contiene informazioni sulle espressioni logiche disponibili e sui tipi di dati per il risultato e gli operandi. Vengono usate le abbreviazioni seguenti:

ANY_BIT per i tipi di dati BOOL, BYTE, WORD, DWORD

Tabella 13-3 Operatori logici

Operazione	Operatore	1º Operando	2º Operando	Risultato	Priorità
Negazione	NOT	ANY_BIT	-	ANY_BIT	3
Congiunzione	AND	ANY_BIT	ANY_BIT	ANY_BIT	8
Disgiunzione esclusiva	XOR	ANY_BIT	ANY_BIT	ANY_BIT	9
Disgiunzione	OR	ANY_BIT	ANY_BIT	ANY_BIT	10

Risultati

Il risultato di un'espressione logica può essere

- 1 (*true*) o = 0 (*false*) in una combinazione di operandi booleani
- oppure una stringa di bit dopo un'operazione logica combinatoria a bit fra i due operandi.

Esempi

Supponiamo che n sia una variabile di numeri interi con il valore 10 e che s sia una variabile di caratteri che rappresenta il carattere A. Seguono alcune espressioni logiche con queste variabili.

```
Espressione                 Valore

 ( n>0 )  AND  ( n<20)      vero
 ( n>0 )  AND  ( n<5 )      falso
 ( n>0 )  OR   ( n<5 )      vero
 ( n>0 )  XOR  ( n<20)      falso
 ( n=10 ) AND  ( s='A')     vero
 ( n<>5 ) OR   ( s>='A')    vero
```

Esempio 13-7 Espressioni logiche

Assegnazione di valori 14

Panoramica

Un'assegnazione di valori serve per assegnare ad una variabile il valore di un'espressione. Il valore attuale viene sovrascritto.

Sommario del capitolo

Ulteriori informazioni

In SCL esistono istruzioni semplici e strutturate. Fanno parte delle istruzioni semplici, oltre all'assegnazione di valore, l'istruzione di richiamo e l'istruzione GOTO. I capitoli 15 e 16 contengono informazioni al riguardo.

Le istruzioni di controllo per diramazioni del programma e per l'elaborazione di loop appartengono alla categoria di istruzioni strutturate. Il capitolo 15 contiene informazioni al riguardo.

14.1 Panoramica

Principio

Le assegnazioni di valori sostituiscono il valore momentaneo di una variabile con un nuovo valore che viene introdotto tramite un'espressione. Questa espressione può contenere anche identificatori di funzioni (FC) che vengono così attivate e ritornano valori corrispondenti (valore di ritorno).

Conformemente al diagramma sintattico 14-1 viene calcolata l'espressione a destra dell'assegnazione di valore e il risultato viene memorizzato nella variabile il cui nome è situato a sinistra del segno di assegnazione. La figura 14-1 illustra l'insieme delle variabili consentite.

Assegnazione di valore

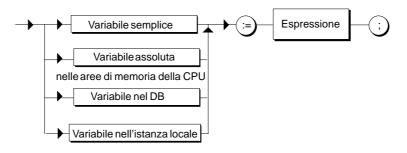



Figura 14-1 Sintassi di un'assegnazione di valore

Risultato

Il tipo di un'espressione di assegnazione è il tipo dell'operando sinistro.

14.2 Assegnazione di valori con variabili di un tipo di dati semplice

Assegnazione

Ogni espressione e ogni variabile con tipo di dati semplici possono essere assegnate ad un'altra variabile dello stesso tipo.

```
Identificatore := Espressione;
Identificatore := Variabile di tipo di dati semplici;
```

Esempi

Alcune assegnazioni di valori valide potrebbero essere:

```
FUNCTION_BLOCK FBx
VAR
      INTERRUT_1  :INT;
      INTERRUT_2  :INT;
      VALRIF_1    :REAL;
      VALRIF_2    :REAL;
      INTERROG_1  :BOOL;
      ORA_1       :S5TIME;
      ORA_2       :TIME;
      DATA_1      :DATE;
      DATGIOR_1   :TIME_OF_DAY;
END_VAR
BEGIN
// Assegnazione di una costante ad una variabile
      INTERRUT_1  := -17;
      INTERRUT_1  := 100.1;
      INTERROG_1  := TRUE;
      ORA_1       :=TIME#1H_20M_10S_30MS;
      ORA_2       :=TIME#2D_1H_20M_10S_30MS;
      DATA_1      :=DATE#1996–01–10;
// Assegnazione di una variabile
      VALRIF_1    := VALRIF_2;
      INTERRUT_2_ := INTERRUT_1;
// Assegnazione di un'espressione ad una variabile
      INTERRUT_2:= INTERRUT_1 * 3;
END_FUNCTION_BLOCK
```

Esempio 14-1 Assegnazione di valori con tipi di dati semplici

14.3 Assegnazione di valori con variabili di tipo STRUCT e UDT

Variabili di tipo STRUCT e UDT

Le variabili di tipo STRUCT e UDT sono variabili strutturate che rappresentano una struttura completa o una componente di questa struttura.

Alcune indicazioni valide per una variabile di struttura potrebbero essere:

```
Figura             //Identificatore per struttura
Figura.elemento    //Identificatore per componente di
                   //struttura
Figura.campo       //Identificatore di un campo semplice
                   //all’interno di una struttura
Figura.campo[2,5]  //Identificatore di una componente di
                   //campo all’interno di una struttura
```

Assegnazione di una struttura completa

Una struttura completa può essere assegnata ad un’altra struttura solo se le componenti della struttura corrispondono sia nel tipo di dati che nel nome. Un’assegnazione valida è:

```
Structname_1:=Structname_2;
```

Assegnazione di componenti di struttura

Ad ogni componente di struttura, può essere assegnata una variabile compatibile al tipo o un’altra componente di struttura. Sono assegnazioni valide:

```
Structname_1.element1      := Valore;
Structname_1.element1      := 20.0;
Structname_1.element1      := Structname_2.element1;
Structname_1.nomedicampo1  := Structname_2.nomedicampo2;
Structname_1.nomedicampo[10]:= 100;
```

Esempi

Gli esempi seguenti illustrano le assegnazioni di valori per i dati delle strutture:

```
VAR
      VALAUSIL: REAL;
      VALMISURA:  STRUCT    //Struttura obiettivo
            TENSIONE:       REAL;
            RESISTENZA:     REAL;
            CAMPOSEMPLICE:  ARRAY[1..2,1..2]OF INT;
      END_STRUCT;

      VALEFFET: STRUCT       //Struttura sorgente
            TENSIONE:       REAL;
            RESISTENZA:     REAL;
            CAMPOSEMPLICE:  ARRAY[1..2,1..2]OF INT;
      END_STRUCT
END_VAR
BEGIN
//Assegnazione di una struttura completa ad
//una struttura completa
      VALMISURA := VALATTUALE;

//Assegnazione di una componente di struttura ad
//una componente di struttura
      VALMISURA.TENSIONE:= VALATTUALE TENSIONE;

// Assegnazione di una componente di struttura ad
// una componente di struttura per una variabile //
dello stesso tipo
      VARAUSIL:= VALATTUALE RESISTENZA;

// Assegnazione di una costante ad una
// componente di struttura
      VALMISURA.RESISTENZA:= 4.5;

// Assegnazione di una costante ad un elemento di
// un campo semplice
      VALMISURA.CAMPOSEMPLICE[1,2]:= 4;
END_FUNCTION_BLOCK
```

Esempio 14-2 Assegnazione di valori con variabili di tipo STRUCT

14.4 Assegnazione di valori con variabili di tipo ARRAY

Variabile di campo

Un campo si compone di 1 fino a max. 6 dimensioni e contiene elementi di campo tutti dello stesso tipo. Per l'assegnazione di campi ad una variabile esistono due varianti di accesso:

Si può fare riferimento a campi **completi** oppure a **campi parziali**. Per fare riferimento ad un campo completo si deve indicare il nome della variabile del campo.

```
nomedicampo_1
```

Un singolo elemento di campo viene indirizzato con il nome del campo seguito da valori indice idonei racchiusi fra parentesi quadre. Per ogni dimensione è disponibile un indice. Essi vengono separati mediante virgole e sono racchiusi fra parentesi quadre. Un indice deve essere un'espressione aritmetica con tipo di dati `INT`.

```
nomedicampo_1[2]
nomedicampo_1[4,5]
```

Assegnazione di un campo completo

Un campo completo può essere assegnato ad un altro campo se corrispondono sia il tipo di dati delle componenti che i limiti di campo (indici di campo più piccoli e più grandi possibili). Sono assegnazioni valide:

```
nomedicampo_1:=nomedicampo_2;
```

Assegnazione di una componente di campo

Per realizzare un'assegnazione di valori per un campo parziale, nelle parentesi quadre si devono omettere indici dietro il nome del campo a partire da destra. In tal modo si indirizza un'area parziale il cui numero di dimensione equivale al numero di indici omessi.

Ne consegue che in una matrice si può fare riferimento a righe e componenti singole chiuse, ma non a colonne chiuse (chiuso significa da ... fino a).

Sono assegnazioni valide:

```
nomedicampo_1[i]:=nomedicampo_2[j];
nomedicampo_1[i]:=Espressione;
identificatore_1 :=nomedicampo_1[i];
```

Esempi

Gli esempi seguenti illustrano le assegnazioni di valori per campi:

```
FUNCTION_BLOCK FB3
VAR
      VALRIF      :ARRAY [0..127] OF INT;
      VALATTUALE  :ARRAY [0..127] OF INT;

// Convenzione per una matrice
// (=campo a due dimensioni)
// con 3 righe e 4 colonne
      REGOLATORE: ARRAY [1..3, 1..4] OF INT;

// Convenzione per un vettore
// (=campo a una dimensione)
// con 4 componenti
      REGOLATORE_1: ARRAY [1..4] OF INT;
END_VAR

BEGIN
// Assegnazione di un campo completo ad un campo

      VALRIF := VALATTUALE;

// Assegnazione di un vettore alla seconda riga del
// campo REGOLATORE
      REGOLATORE [2]:= REGOLATORE_1;

//Assegnazione di una componente di campo
//ad una componente del campo REGOLATORE
      REGOLATORE [1,4]:= REGOLATORE_1 [4];
END_FUNCTION_BLOCK
```

Esempio 14-3 Assegnazione di valori per campi

14.5 Assegnazione di valori con variabili di tipo STRING

Variabile STRING

Una variabile con tipo di dati STRING contiene una stringa di caratteri contenente max. 254 caratteri.

Assegnazione

Ad ogni variabile con tipo di dati STRING si può assegnare un'altra variabile dello stesso tipo. Sono assegnazioni valide:

```
variabilestringa_1 := literalstringa ;
variabilestringa_1 := variabilestringa_2 ;
```

Esempio

Gli esempi seguenti illustrano le assegnazioni di valori con variabili STRING:

```
FUNCTION_BLOCK FB3
VAR
      VISUALIZ_1  : STRING[50] ;
      STRUTTURA1  : STRUCT
              VISUALIZ_2 : STRING[100] ;
              VISUALIZ_3 : STRING[50] ;
      END_STRUCT;
END_VAR
BEGIN
// Assegnazione di una costante ad una
// variabile STRING

      VISUALIZ_1 := ’Errore nel modulo 1’ ;

// Assegnazione di una componente di struttura
// ad una variabile STRING.

      VISUALIZ_1 := STRUTTURA1.VISUALIZ_3 ;

// Assegnazione di una variabile STRING ad
// una componente di struttura STRING

      If VISUALIZ_1 <> STRUTTURA.VISUALIZ_3 THEN
        VISUALIZ_1 := STRUTTURA.VISUALIZ_3 ;
      END_IF;
END_FUNCTION_BLOCK
```

Esempio 14-4 Assegnazione di valori per variabili STRING

14.6 Assegnazione di valori con variabili di tipo DATE_AND_TIME

Variabile DATE_AND_TIME

Il tipo di dati DATE_AND_TIME definisce un'area di 64 bit (8 byte) per l'indicazione della data e dell'ora.

Assegnazione

Ad ogni variabile con tipo di dati DATE_AND_TIME si può assegnare un'altra variabile o costante dello stesso tipo. Sono assegnazioni valide:

```
dtvariable_1 := Data e literal tempo ;
dtvariable_1 := dtvariable_2 ;
```

Esempio

Gli esempi seguenti illustrano le assegnazioni di valori con variabili DATE_AND_TIME:

```
FUNCTION_BLOCK FB3
VAR
      ORA_1       : DATE_AND_TIME;
      STRUTTURA1  : STRUCT
              ORA_2 : DATE_AND_TIME ;
              ORA_3: DATE_AND_TIME ;
              END_STRUCT;
END_VAR
BEGIN
// Assegnazione di una costante per una
// variabile DATE_AND_TIME
ORA_1 := DATE_AND_TIME#1995-01-01-12:12:12.2 ;
STRUTTURA.ORA_3 := DT#1995-02-02-11:11:11 ;

// Assegnazione di una componente di struttura per
// una variabile DATE_AND_TIME.

ORA_1 := STRUTTURA1.ORA_2 ;


// Assegnazione di una variabile DATE_AND_TIME
//  ad una componente di struttura DATE_AND_TIME
If ORA_1 < STRUTTURA1.ORA_3 THEN
   STRUTTURA1.ORA_3 := ORA_1 ;
END_IF;
END_FUNCTION_BLOCK
```

Esempio 14-5 Assegnazione di valori con variabili di tipo DATE_AND_TIME

14.7 Assegnazione di valori con variabili assolute per aree di memoria

Variabile assoluta

La variabile assoluta fa riferimento alle aree di memoria valide della CPU. Per l'assegnazione di valori si hanno le tre possibilità di indirizzamento descritte nel capitolo 12.

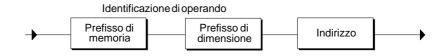



Figura 14-2 Sintassi per la variabile assoluta

Assegnazione

Ad eccezione degli ingressi di periferia e degli ingressi tramite l'immagine di processo, ad ogni variabile assoluta si può assegnare una variabile o un'espressione dello stesso tipo.

Esempio

Gli esempi seguenti illustrano le assegnazioni di valori per variabili assolute:

```
VAR
      PAROLADISTATO1: WORD;
      PAROLADISTATO2: BOOL;
      PAROLADISTATO3: BYTE;
      PAROLADISTATO4: BOOL;
      INDIRIZZO: INT:= 10;
END_VAR
BEGIN
      // Assegnazione di una parola d'ingresso
      // ad una variabile (accesso semplice)
      PAROLADISTATO1:= EW4 ;

      // Assegnazione di un bit di uscita
      // ad una variabile (accesso semplice)
      PAROLADISTATO2:= a1.1 ;

      // Assegnazione di un byte d'ingresso ad
      // una variabile (accesso indicizzato)
      PAROLADISTATO3:= EB[INDIRIZZO];

      // Assegnazione di un bit d'ingresso ad
      // una variabile (accesso indicizzato)
      FOR INDIRIZZO:= 0 TO 7 BY 1 DO
      PAROLADISTATO4:= e[1,INDIRIZZO] ;
      END_FOR;
```

Esempio 14-6 Assegnazione di valori per variabili assolute

14.8 Assegnazione di valori con variabili globali

Variabili nei DB

Anche l'accesso alle variabili globali nei blocchi dati avviene tramite un'assegnazione di valori a variabili dello stesso tipo o viceversa. Si ha la possibilità di accedere in modo strutturato, assoluto o indicizzato (capitolo 12).

Variabile DB

Figura 14-3 Sintassi di una variabile DB

Assegnazione

Ad ogni variabile globale si può assegnare una variabile o un'espressione dello stesso tipo. Seguono alcune assegnazioni valide:

```
DB11.DW10:=20;
DB11.DW10:=stato;
```

Esempi

L'esempio seguente presuppone che nel blocco dati DB 11 siano state definite una variabile ″NUMERO″ con tipo di dati INTEGER e una struttura ″NUMERO1″ con la componente ″NUMERO2″ del tipo INTEGER.

```
// Blocco dati DB 11 necessario
DATA_BLOCK DB 11
STRUCT
      NUMERO:     INT:=1;
      NUMERO1:    STRUCT
      NUMERO2:    INT := 256;
END_STRUCT;
PAROLA3:    WORD:=W#16#aa;
PAROLA4:    WORD:=W#16#aa;
PAROLA5:    WORD:=W#16#aa;
PAROLA6:    WORD:=W#16#aa;
PAROLA7:    WORD:=W#16#aa;
PAROLA8:    WORD:=W#16#aa;
PAROLA9:    WORD:=W#16#aa;
PAROLA10:   WORD;
END_STRUCT

BEGIN
PAROLA10:=W#16#bb;
END_DATA_BLOCK
```

Esempio 14-7 Assegnazione di valori a variabili globali

Per esempio, il blocco dati DB 11 può essere impiegato come segue:

```
VAR
      REGOLATORE_1: ARRAY [1..4] OF INT;
      PAROLADISTATO1: WORD ;
      PAROLADISTATO2: ARRAY [1..4] OF INT;
      PAROLADISTATO3: INT ;
      INDIRIZZO     : INT ;
END_VAR
BEGIN
      // Assegnazione della parola 10 da DB 11
      // a una variabile (accesso semplice)
      PAROLADISTATO1:= DB11.DW10

      // Alla prima componente di campo
      // viene assegnata la variabile
      // ”NUMERO” da DB 11(accesso strutturato):
      REGOLATORE_1[1]:= DB11.ZAHL;

      // Assegnazione della componente di strut-
      // tura ”NUMERO2” della struttura ”NUMERO1”
      // alla variabile PAROLADISTATO3
      PAROLADISTATO3:= DB11.NUMERO1.NUMERO2

      // Assegnazione di una parola con
      // indice INDIRIZZO da DB11 ad una variabile
      // (accesso indicizzato)
      FOR INDIRIZZO:= 1 TO 10 BY 1 DO
      PAROLADISTATO2[ADRESSE]:= DB11.DW[INDIRIZZO];
      END_FOR;
```

Esempio 14-8 Assegnazione di valori alle variabili locali di un DB

15 Istruzioni di controllo

Panoramica

Si possono programmare solo pochi blocchi in modo che tutte le istruzioni fino alla fine del blocco vengono eseguite una dopo l'altra. In genere, in funzione delle condizioni indicate, vengono eseguite solo determinate istruzioni (alternative) oppure vengono ripetute più volte (loop). A tal fine, i mezzi tecnici di programmazione sono le istruzioni di controllo all'interno di un blocco SCL.

Sommario del capitolo

15.1 Panoramica

Istruzioni di selezione

Nei programmi, spesso si verifica il problema di dover eseguire determinate sequenze di istruzioni in funzione di certe condizioni. La diramazione di programma (Selezione) offre la possibilità di controllare il flusso di programma attivando da 2 fino a n sequenze di istruzioni alternative.

Tabella 15-1 Tipi di diramazioni

Tipo di diramazione	Funzione
Istruzione `IF`	Con l'istruzione IF si può far proseguire l'esecuzione del programma da una diramazione scelta fra due alternative in funzione di un determinata condizione che può essere TRUE o FALSE.
Istruzione `CASE`	Con l'istruzione CASE si può far proseguire l'esecuzione del programma in base ad una diramazione di tipo 1:n, assegnando ad una variabile un determianto valore scelto fra n valori possibili.

Istruzioni di ripetizione

L'elaborazione di loop può essere controllata con l'ausilio di istruzioni di ripetizione. Un'istruzione di ripetizione indica quali parti di un programma devono essere ripetute in funzione di determinate condizioni.

Tabella 15-2 Tipi di istruzioni per l'elaborazione di loop

Tipo di diramazione	Funzione
Istruzione `FOR`	serve per ripetere una sequenza di istruzioni finché la variabile d'esecuzione rimane all'interno del campo di valori indicato.
Istruzione `WHILE`	serve per ripetere una sequenza di istruzioni finché è soddisfatta una determinata condizione d'esecuzione.
Istruzione `REPEAT`	serve per ripetere una sequenza di istruzioni finché non venga soddisfatta una determinata condizione d'interruzione.

Istruzioni di salto

Un salto di programma causa il salto immediato a una determinata etichetta di salto e quindi ad un'istruzione all'interno dello stesso blocco.

Tabella 15-3 Tipi di istruzioni di salto

Tipo di diramazione	Funzione
Istruzione `CONTINUE`	serve per interrompere la momentanea esecuzione del loop.
Istruzione `EXIT`	serve per uscire da un loop in qualsiasi momento e indipendentemente dal fatto che sia soddisfatta o non una determinata condizione d'interruzione.
Istruzione `GOTO`	causa il salto immediato ad un'etichetta di salto indicata.
Istruzione `RETURN`	causa l'uscita da un blocco momentaneamente in corso di elaborazione.

Condizioni

La condizione può essere un'espressione di confronto, o un'espressione logica. Essa è del tipo BOOL e può assumere i valori TRUE o FALSE.

Seguono alcuni esempi di valide **espressioni di confronto**:

```
CONTATORE<=100
SQR(A)>0.005
Risposta = 0
SALDO>=RIPORTO
ch1< 'T'
```

Seguono alcuni esempi per l'impiego di espressioni di confronto con operatori **logici**:

```
(CONTATORE<=100) AND(CH1<'*')
(SALDO<100.0) OR (STATUS ='R')
(Risposta<0)OR((Risposta>5.0) AND (RISPOSTA<10.0))
```

Avvertenza

Si osservi che gli operandi logici (qui espressioni di confronto) sono riportati fra parentesi per evitare diverse interpretazioni sulla sequenza durante la valutazione.

15.2 Istruzione IF

Principio

L'istruzione IF è un'istruzione condizionata. Essa consente una o più opzioni e seleziona una delle sue parti istruzioni (se necessario nessuna) per l'esecuzione.

Istruzione IF

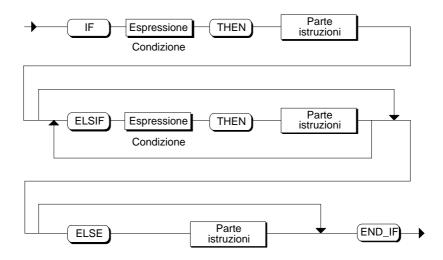



Figura 15-1 Sintassi dell'istruzione IF

L'esecuzione delle istruzioni condizionate causa la valutazione dell'espressione logica indicata. Se il valore di un'espressione è TRUE, la condizione viene considerata soddisfatta, se il valore è FALSE la condizione non è soddisfatta.

Esecuzione

L'istruzione IF viene elaborata in base alle seguenti regole:

1. Se il valore della prima espressione è TRUE, viene eseguita la parte istruzioni che segue `THEN`, altrimenti vengono valutate le espressioni contenute nei rami `ELSIF`.
2. Se nei rami `ELSIF` non è presente alcuna espressione blooleana TRUE, viene eseguita la sequenza di istruzioni di ELSE (o non viene eseguita alcuna sequenza di istruzioni se non è presente alcun ramo `ELSE`).

Vi può essere un numero qualunque di istruzioni ELSIF.

Si osservi che i rami `ELSIF` e / o il ramo `ELSE` possono essere assenti. Questi casi vengono trattati come se questi rami contenessero istruzioni vuote.

Avvertenza

Si osservi che l'istruzione `END_IF` deve essere conclusa con un punto e virgola**.**

Avvertenza

Rispetto ad una sequenza di istruzioni IF, l'impiego di uno o più rami ELSIF presenta il vantaggio che le espressioni logiche che seguono un'espressione valida non vengono più calcolate. Ciò consente di ridurre il tempo d'esecuzione di un programma.

Esempio

L'esempio 15-1 illustra l'uso dell'istruzione IF:

```
IF E1.1 THEN
      N:= 0;
      SUM:= 0;
      OK:= FALSE; // Attiva OK-Flag su FALSE
ELSIF START = TRUE THEN
      N:= N + 1;
      SUM:= SUM + N;
ELSE
      OK:= FALSE;
END_IF;
```

Esempio 15-1 Istruzioni IF

15.3 Istruzione CASE

Principio

L'istruzione CASE serve per la selezione di 1 da n di una parte di programma. Questa selezione si basa sul valore corrente di un'espressione di selezione.

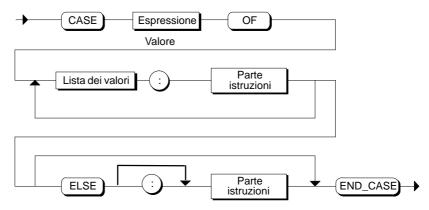



Figura 15-2 Sintassi dell'istruzione CASE

Esecuzione

L'esecuzione dell'istruzione CASE avviene conformemente alle regole seguenti:

1. Durante l'elaborazione dell'istruzione `CASE` si verifica se il valore dell'espressione di selezione è contenuto in una determinata lista di valori. Ogni valore di questa lista rappresenta uno dei valori consentiti per l'espressione di selezione. L'espressione di selezione deve fornire un valore del tipo INTEGER.
2. Se il confronto ha esito positivo, viene eseguita la parte istruzioni assegnata alla lista.
3. Il ramo ELSE è opzionale e viene eseguito se il confronto non ha esito positivo.

Avvertenza

Si osservi che l'istruzione `END_CASE` deve essere conclusa con un punto e virgola**.**

Lista dei valori

Essa contiene i valori consentiti per l'espressione di selezione.

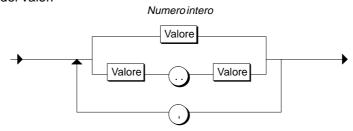



Figura 15-3 Sintassi della lista dei valori

Regole

Durante la formulazione della lista dei valori si deve osservare quanto segue:

- Ogni lista di valori può iniziare con una costante, una lista di costanti o un'area di costanti,
- I valori all'interno della lista di valori devono essere del tipo INTEGER,
- Ogni valore può essere presente una sola volta,

Esempi

L'esempio 15-2 illustra l'uso dell'istruzione CASE. La variabile TW è di regola del tipo INTEGER.

```
CASE TW OF
      1:    DISPLAY   := OVEN_TEMP;
      2:    DISPLAY   := MOTOR_SPEED;
      3:    DISPLAY   := GROSS_TARE;
                  AW4 := 16#0003;
      4..10:DISPLAY   := INT_TO_DINT (TW);
                  AW4 := 16#0004;
      11,13,19:DISPLAY:= 99;
                  AW4 := 16#0005;
ELSE:       DISPLAY   := 0;
            TW_ERROR  := 1;
END_CASE;
```

Esempio 15-2 Istruzione CASE

Avvertenza

Si osservi che il tempo d'esecuzione dei loop non deve essere troppo lungo altrimenti la CPU emette un messaggio di ritardo alla conferma e si porta in STOP.

15.4 Istruzione FOR

Principio

L'istruzione FOR o istruzione di ripetizione esegue una sequenza di istruzioni in un loop in cui ad una variabile di esecuzione vengono assegnati valori in ordine progressivo. La variabile d'esecuzione deve essere l'identificatore di una variabile del tipo INT o DINT.

Sintassi

Istruzione FOR

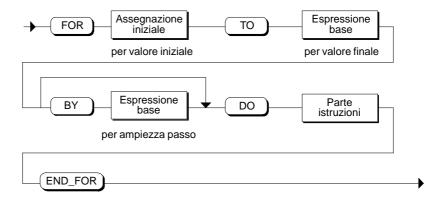



Figura 15-4 Sintassi dell'istruzione FOR

La definizione di un loop con FOR comprende anche la definizione di un valore iniziale e di un valore finale. Entrambi i valori devono essere dello stesso tipo della variabile d'esecuzione.

Esecuzione

L'istruzione FOR viene elaborata conformemente alle regole seguenti:

1. Quando viene attivato il loop, la variabile d'esecuzione viene impostata sul valore iniziale e ad ogni passaggio del loop viene incrementata dell'ampiezza di passo indicata (ampiezza di passo positiva) o diminuita (ampiezza di passo negativa) finché non venga raggiunto il valore finale.
2. Dopo ogni giro si verifica se la condizione

   `|valore iniziale| <= |valore finale|`

   è soddisfatta. In caso positivo, la sequenza di istruzioni viene eseguita, mentre in caso negativo il loop, e quindi la sequenza di istruzioni, viene saltata.

Avvertenza

Si osservi che l'istruzione `END_FOR` deve essere conclusa con un punto e virgola.

Assegnazione iniziale

Per la formazione del valore iniziale della variabile d'esecuzione è disponibile l'assegnazione iniziale rappresentata in figura 15-5.

Assegnazione iniziale

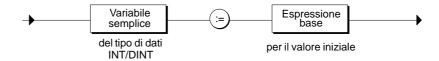



Figura 15-5 Sintassi per l'assegnazione iniziale

Esempi:

```
FOR I  := 1 TO 20
FOR I  := 1 TO (Inizio+J)
```

Valore finale e ampiezza di passo

Per la formazione del valore finale e della ampiezza di passo desiderata si può formare un'espressione base.

Regole

Valgono le seguenti regole per l'istruzione FOR:

- L'indicazione di BY [*ampiezza di passo*] può essere omessa. Se l'ampiezza di passo non viene specificata, essa assume il valore `+1`.
- Il valore iniziale, il valore finale e l'ampiezza di passo sono espressioni (vedere capitolo 13). L'analisi avviene una sola volta all'inizio dell'esecuzione dell'istruzione `FOR`.
- Durante l'esecuzione del loop non è consentita alcuna modifica del valore finale e dell'ampiezza di passo.

Esempio

L'esempio 15-3 illustra l'uso dell'istruzione FOR:

```
FUNCTION_BLOCK RICERCA
VAR
      INDICE      : INT;
      PAROLAIDENT : ARRAY [1..50] OF STRING;
END_VAR

BEGIN
FOR INDICE:= 1 TO 50 BY 2 DO
      IF PAROLAIDENT [INDEX] = 'KEY' THEN
      EXIT;
      END_IF;
END_FOR;
END_FUNCTION_BLOCK
```

Esempio 15-3 Istruzione FOR

15.5 Istruzione WHILE

Principio

L'istruzione WHILE consente l'esecuzione iterativa di una sequenza di istruzioni sotto il controllo di una condizione d'esecuzione. La condizione d'esecuzione viene generata in base alle regole di una espressione logica.

Sintassi

Istruzione WHILE

Figura 15-6 Sintassi dell'istruzione WHILE

La parte istruzioni che segue `DO` viene ripetuta finché la condizione d'esecuzione possiede il valore TRUE.

Esecuzione

L'istruzione `WHILE` viene elaborata conformemente alle regole seguenti:

1. La condizione d'esecuzione viene analizzata **prima** di ogni esecuzione della parte istruzioni.
2. Se si verifica il valore TRUE, viene eseguita la parte istruzioni.
3. Se si verifica il valore FALSE, l'esecuzione dell'istruzione WHILE viene conclusa. Ciò può verificarsi fin dalla prima analisi del risultato.

Avvertenza

Si osservi che l'istruzione `END_WHILE` deve essere conclusa con un punto e virgola.

Esempio

L'esempio 15-4 illustra l'uso dell'istruzione `WHILE`:

```
FUNCTION_BLOCK RICERCA
VAR
      INDICE      : INT;
      PAROLAIDENT : ARRAY [1..50] OF STRING;
END_VAR
BEGIN
INDICE:= 1;
WHILE INDICE <= 50 AND PAROLAIDENT[INDEX]<>'KEY' DO
      INDICE:= INDICE + 2;
END_WHILE;
END_FUNCTION_BLOCK
```

Esempio 15-4 Istruzione WHILE

15.6 Istruzione REPEAT

Principio

Un'istruzione REPEAT causa l'esecuzione iterativa di una sequenza di istruzioni compresa fra REPEAT e UNTIL finché non si verifica una condizione di interruzione. La condizione di interruzione viene generata in base alle regole di una espressione logica.

Istruzione REPEAT

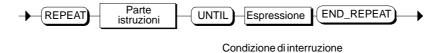



Figura 15-7 Sintassi dell'istruzione REPEAT

La condizione viene verificata **dopo** l'esecuzione della sequenza delle istruzioni. Ciò significa che il nucleo delle istruzioni viene eseguito almeno una volta, anche se la condizione di interruzione è soddisfatta fin dall'inizio.

Avvertenza

Si osservi che l'istruzione `END_REPEAT` deve essere conclusa con un punto e virgola.

Esempio

L'esempio 15-5 illustra l'uso dell'istruzione REPEAT.

```
FUNCTION_BLOCK RICERCA
VAR
      INDICE      : INT;
      PAROLAIDENT : ARRAY [1..50] OF STRING;
END_VAR

BEGIN
INDICE:= 0;
REPEAT
      INDICE:= INDICE + 2;
UNTIL
      INDICE > 50 OR PAROLAIDENT[INDEX] = 'KEY'
END_REPEAT;
END_FUNCTION_BLOCK
```

Esempio 15-5 Istruzione REPEAT

15.7 Istruzione CONTINUE

Principio

L'istruzione CONTINUE serve per interrompere l'esecuzione momentanea di un loop di un'istruzione di ripetizione (FOR, WHILE o REPEAT) e per riavviare l'esecuzione all'interno di un loop.

Istruzione CONTINUE

Figura 15-8 Sintassi dell'istruzione CONTINUE

Se una sequenza di istruzioni deve essere ripetuta o viene deciso in un loop WHILE dalla condizione iniziale e in un loop REPEAT dalla condizione finale.

In un'istruzione FOR, la variabile d'esecuzione viene incrementata dell'ampiezza di passo indicata direttamente dopo un'istruzione CONTINUE.

Esempio

L'esempio 15-6 illustra l'uso dell'struzione CONTINUE:

```
FUNCTION_BLOCK_CONTINUE
VAR
      INDICE      :INT;
      CAMPO       :ARRAY[1..100] OF INT;
END_VAR
BEGIN
INDEX:= 0;
WHILE INDICE <= 100 DO
      INDICE:= INDICE + 1;
      // Se CAMPO[INDICE] è uguale a INDICE,
      // CAMPO [INDICE] non viene modificato:
            IF CAMPO[INDICE] = INDICE THEN
                  CONTINUE;
            END_IF;
            CAMPO[INDICE]:= 0;
      // Ulteriori istruzioni..
      //....
END_WHILE;
END_FUNCTION_BLOCK
```

Esempio 15-6 Istruzione CONTINUE

15.8 Istruzione EXIT

Principio

Un'istruzione **EXIT** serve per uscire da un loop (`FOR, WHILE` o `REPEAT`) in qualsiasi momento e indipendentemente dal fatto se la condizione di interruzione sia stata soddisfatta o non.

Istruzione EXIT

Figura 15-9 Sintassi dell'istruzione EXIT

Questa istruzione causa l'uscita immediata da quella istruzione di ripetizione che circonda immediatamente l'istruzione EXIT.

L'esecuzione del programma viene continuata dopo la fine del loop di ripetizione (p. es. dopo `END_FOR`).

Esempio

L'esempio 15-7 illustra l'uso dell'istruzione EXIT:

```
FUNCTION_BLOCK_EXIT
VAR
      INDICE_1       := INT;
      INDICE_2       := INT;
      INDICE_CERCATO := INT;
      PAROLAIDENT    : ARRAY[1..51] OF STRING;
END_VAR
BEGIN
INDICE_2    := 0;
FOR INDICE_1:= 1 TO 51 BY 2 DO
      // Uscita dal loop FOR se
      // PAROLAIDENT[INDICE_1] è uguale a 'KEY':
      IF PAROLAIDENT[INDICE_1] = 'KEY' THEN
            INDICE_2:= INDICE_1;
            EXIT;
      END_IF;
END_FOR;
// La seguente assegnazione di valore viene
// eseguita dopo l'esecuzione di EXIT o dopo la
// regolare esecuzione del loop FOR:
INDICE_CERCATO:= INDICE_2;
END_FUNCTION_BLOCK
```

Esempio 15-7 Istruzione EXIT

15.9 Istruzione GOTO

Principio

Con una istruzione GOTO si può realizzare un salto di programma. Essa causa il salto immediato ad un'etichetta di salto indicata e quindi ad un'altra istruzione all'interno dello stesso blocco.

Le istruzioni GOTO dovrebbero essere utilizzate solo in casi speciali, p. es. per il trattamento degli errori. In base alle regole della programmazione strutturata, l'istruzione GOTO non dovrebbe essere utilizzata affatto.

Salto di programma

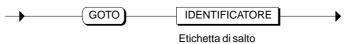



Figura 15-10 Sintassi dell'istruzione GOTO

In questo caso l'etichetta di salto indica un'etichetta nel blocco dichiarazioni `LABEL/END_LABEL`. Questa etichetta precede l'istruzione che deve essere eseguita per prima dopo `GOTO`.

Regole

Quando si impiega l'istruzione GOTO si devono osservare le regole seguenti:

- La destinazione di un'istruzione di salto deve trovarsi all'interno dello stesso blocco.
- La destinazione di salto deve essere univoca.
- Un salto in un blocco di loop non è consentito. È possibile, invece, un salto da un blocco di loop.

Esempio

L'esempio 15-8 illustra l'uso dell'istruzione GOTO:

```
FUNCTION_BLOCK FB3//GOTO_BSP
VAR
      INDICE  : INT;
      A       : INT;
      B       : INT;
      C       : INT;
      PAROLA DI IDENTIFICAZIONE: ARRAY[1..51] OF STRING;
END_VAR
LABEL
      ETICHETTA1, ETICHETTA 2, ETICHETTA 3;
END_LABEL
BEGIN
IF A > B THEN GOTO ETICHETTA 1;
      ELSIF A > C THEN GOTO ETICHETTA 2;
END_IF;
//...
ETICHETTA 1 :     INDICE:= 1;
                  GOTO ETICHETTA3;
ETICHETTA 2 :     INDICE:= 2;
//...
ETICHETTA 3 :     ;
//...
```

Esempio 15-8 Istruzione di salto GOTO

15.10 Istruzione RETURN

Principio

Un’istruzione RETURN causa l’uscita dal blocco momentaneamente in elaborazione (OB, FB, FC) e il ritorno al blocco richiamante o al sistema operativo quando si esce da un OB.

Istruzione RETURN

Figura 15-11 Sintassi dell’istruzione RETURN

Avvertenza

Un’istruzione RETURN alla fine della parte di esecuzione di un blocco di codice o della parte dichiarazione di un blocco dati è ridondante poiché essa viene eseguita automaticamente.

Richiamo di funzioni e blocchi funzionali 16

Panoramica

Da un blocco SCL si possono richiamare quali funzioni (FC) o blocchi funzionali (FB):

- Funzioni e blocchi funzionali generati in SCL.
- Funzioni e blocchi funzionali che sono stati programmati in un altro linguaggio STEP 7 (p. es. AWL, KOP).
- Funzioni di sistema (SFC) e blocchi funzionali di sistema (SFB) disponibili nel sistema operativo della CPU usata dall'utente.

Sommario del capitolo

16.1 Richiamo e trasferimento di parametri

Trasferimento di parametri

Durante il richiamo di funzioni o blocchi funzionali avviene uno scambio di dati fra il blocco richiamante e richiamato. I parametri che devono essere trasferiti devono essere indicati nel richiamo come lista di parametri. I parametri vengono inseriti tra parentesi. Più parametri vengono separati mediante virgole.

Principio

Nel seguente esempio di un richiamo di una funzione vengono indicati un parametro d’ingresso, un parametro di transito e un parametro d’uscita.

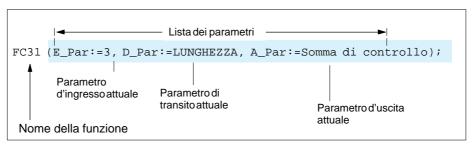



Figura 16-1 Principio del trasferimento di parametri

L’indicazione dei parametri avviene in forma di assegnazione di valori (vedere figura 16-2). Mediante questa assegnazione di valori si assegnano determinati valori (parametri attuali) ai vari parametri che sono stati definiti nella parte dichiarazioni del blocco richiamato (parametri formali).

Parametri attuali		Parametri formali
`3`	⇒×	`E_Par`
`LUNGHEZZA`	⇔	`D_Par`
`Somma di controllo`	⇐	`A_Par`

Figura 16-2 Assegnazione di valori all’interno della lista dei parametri

Parametri formali

I parametri formali sono parametri che il blocco aspetta di ricevere durante il richiamo. Essi sono semplicemente dei ”Segnaposti” per i parametri attuali che vengono trasferiti al blocco durante il richiamo. Questi parametri sono stati definiti dall’utente nella parte dichiarazioni di un blocco (FB o FC).

Tabella 16-1 Blocchi di dichiarazioni consentiti per parametri formali

Blocchi di dichiarazione	Dati	Parola chiave

Blocco parametri	Parametro d’ingresso	`VAR_INPUT`
		Lista dichiarazioni
		`END_VAR`

	Parametro d’uscita	`VAR_OUTPUT`
		Lista dichiarazioni
		`END_VAR`

	Parametro di transito	`VAR_IN_OUT`
		Lista dichiarazioni
		`END_VAR`

16.2 Richiamo di blocchi funzionali (FB o SFB)

Istanza globale e locale

Durante il richiamo di un blocco funzionale, in SCL si possono utilizzare

- sia blocchi dati di istanza globali
- sia aree di istanza locale del blocco dati di istanza attuale.

Il richiamo di un FB come istanza locale si distingue dal richiamo come istanza globale per il modo in cui vengono memorizzati i dati. In questo caso i dati non vengono depositati in un DB separato, ma nel blocco dati di istanza dell'FB richiamante.

Richiami di FB

FB: Blocco funzionale
SFB: Blocco funzionale di sistema

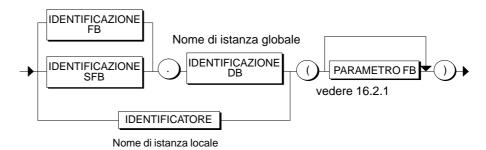



Figura 16-3 Sintassi del richiamo di FB

Richiamo come istanza globale

Il richiamo avviene in un'istruzione di richiamo con indicazione:

- del nome del blocco funzionale o del blocco funzionale di sistema (identificazione di FB o SFB)
- del blocco dati di istanza (identificazione DB)
- nonché dell'assegnazione dei parametri (parametri FB).

Un richiamo di un'istanza globale può essere definito in modo assoluto o simbolico.

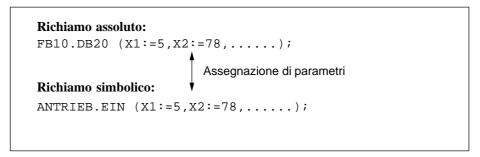



Figura 16-4 Richiamo di FB10 con blocco dati di istanza DB20

Richiamo come istanza locale

Il richiamo avviene in un'istruzione di richiamo con indicazione:

- del nome di istanza locale (IDENTIFICATORE)
- nonché dell'assegnazione dei parametri (parametri FB).

Un richiamo di un'istanza locale è sempre simbolico, p. es.:

```
MOTORE (X1:=5,X2:=78,......);
```

Alimentazione dei parametri

Figura 16-5 Richiamo di un'istanza locale

16.2.1 Parametri FB

Principio

Nel richiamo di un blocco funzionale come istanza globale o locale, nella lista parametri si deve distinguere fra:

- i parametri d'ingresso e
- i parametri di transito

di un FB. In entrambi i casi, mediante **assegnazioni di valori** si assegnano parametri attuali ai parametri formali:

Parametri formali		Parametri attuali
E_Par	⇐	3 //Assegnazione d'ingresso
D_Par	⇐	LUNGHEZZA //Assegnazione di transito

Figura 16-6 Assegnazione di valori nella lista dei parametri

I parametri di uscita non vengono indicati durante il richiamo di un FB, bensì dopo il richiamo di esso.

La sintassi dell'indicazione dei parametri di FB è identica per istanze globali e locali.

Parametri FB

Figura 16-7 Sintassi per i parametri di FB

Esempio

Un richiamo con un'assegnazione di un parametro d'ingresso e di un parametro di transito potrebbe avere p. es. l'aspetto seguente:

```
FB31.DB77(E_Par:=3, D_Par:=LUNGHEZZA);
```

Regole

Per l'alimentazione dei parametri valgono le regole seguenti:

- La sequenza delle assegnazioni è irrilevante.
- I tipi di dati dei parametri formali e attuali devono coincidere.
- Le singole assegnazioni sono separate fra loro mediante virgole.
- Le assegnazioni d'uscita non sono consentite nei richiami di FB. Il valore di un parametro d'uscita definito in precedenza è memorizzato nei dati di istanza. Ad esso si può accedere da qualsiasi FB. Per poterlo leggere da un FB, l'utente deve definire un apposito accesso (vedere capitolo 14.8).

Risultato dopo il richiamo

Dopo l'esecuzione del blocco:

- i parametri d'ingresso attuali trasferiti sono immutati
- i parametri d'ingresso di transito trasferiti ma modificati sono aggiornati. Fanno eccezione i parametri di transito di un tipo di dati semplici (vedere in tal senso il capitolo 16.2.3)
- i parametri d'uscita del blocco richiamante possono essere letti dal blocco dati di istanza locale. Vedere in tal senso l'esempio 16-3.

16.2.2 Assegnazione d'ingresso (FB)

Principio

Mediante assegnazioni d'ingresso, vengono assegnati ai parametri d'ingresso formali dei parametri attuali. L'FB **non** può modificare questi parametri attuali. L'assegnazione di parametri d'ingresso attuali è opzionale. Se non viene indicato alcun parametro attuale, i valori dell'ultimo richiamo rimangono immutati.

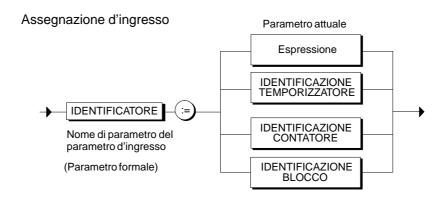



Figura 16-8 Sintassi dell'assegnazione d'ingresso

Possibili parametri attuali

Nelle assegnazioni d'ingresso sono possibili i seguenti parametri attuali:

Tabella 16-2 Parametri attuali nelle assegnazioni d'ingresso

Parametri attuali	Spiegazione
Espressione	• Espressione aritmetica, logica o di confronto • Costante • Variabile ampliata
Identificazione TEMPORIZZATORE /CONTATORE	Definiscono un determinato temporizzatore o contatore, il quale viene utilizzato per l'elaborazione di un blocco (vedere anche capitolo 17).
Identificazione del BLOCCO	Definiscono un determinato blocco, il quale viene utilizzato come parametro d'ingresso. Il tipo di blocco (FB, FC, DB) viene definito nella dichiarazione del parametro d'ingresso. Nell'assegnazione di parametri si deve indicare il numero del blocco. Si può indicare sia il numero assoluto che quello simbolico (vedere anche capitolo 9).

16.2.3 Assegnazione di transito (FB)

Principio

Le assegnazioni di transito vengono usate per assegnare parametri attuali ai parametri di transito formali dell'FB richiamato.

Contrariamente al parametro di ingresso, l'FB richiamante può modificare i parametri di transito. Il nuovo valore del parametro creato durante l'elaborazione dell'FB viene registrato nuovamente nel parametro attuale. Il valore originale viene sovrascritto.

Se nell'FB richiamato sono stati definiti dei parametri di transito, questi devono essere alimentati durante il primo richiamo. Nelle esecuzioni successive l'indicazione dei parametri attuali è opzionale.

Assegnazione di transito

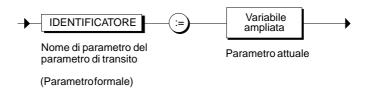



Figura 16-9 Sintassi dell'assegnazione di transito

Parametri attuali di un'assegnazione di transito

Poiché il parametro attuale assegnato durante l'esecuzione di un FB può essere modificato come parametro di transito, esso deve essere una variabile. Per tale motivo, nelle assegnazioni di transito un parametro d'ingresso non può essere assegnato (non sarebbe possibile attualizzare il nuovo valore).

Tabella 16-3 Parametri attuali nelle assegnazioni di transito

Parametro attuale	Spiegazione
Variabile ampliata	Sono disponibili i seguenti tipi di variabili ampliate: variabili e parametri semplici accesso a variabili assolute accesso a blocchi dati richiamo di funzioni (vedere anche capitolo 14).

Particolarità

Si osservino le seguenti particolarità:

- Dopo l'esecuzione del blocco, il valore modificato del parametro di transito viene aggiornato. Fanno eccezione i parametri di transito di un tipo di dati **semplice**. In questo caso, l'aggiornamento avviene solo se nel richiamo è stato indicato un parametro attuale.
- Nei parametri di transito di un tipo di dati **non semplici**, non sono consentiti come parametri attuali:
  - i parametri di transito di FB
  - i parametri di FC
- **Parametro ANY:** in linea di massima, anche qui valgono le prime due dichiarazioni. Inoltre, le costanti **non** sono consentite come parametri attuali.

16.2.4 Esempio di richiamo di un'istanza globale

Principio

Un blocco funzionale con un loop FOR potrebbe avere l'aspetto seguente (vedere l'esempio 16-1). Negli esempi riportati si presuppone che nella tabella dei simboli per FB17 sia stato definito il simbolo TEST.

```
FUNCTION_BLOCK TEST
      VAR_INPUT
            VALORE FINALE: INT; //Parametro d'ingresso
      END_VAR
      VAR_IN_OUT
            IQ1: REAL; //Parametro di transito
      END_VAR
      VAR_OUTPUT
            CONTROL: BOOL;//Parametro d'uscita
      END_VAR
      VAR
            INDEX: INT;
      END_VAR
      BEGIN
            CONTROL:= FALSE;
            FOR INDEX:= 1 TO VALORE FINALE DO
            IQ1:= IQ1 * 2;
            IF IQ1 > 10000 THEN
            CONTROL:= TRUE;
            END_IF;
            END_FOR;
END_FUNCTION_BLOCK
```

Esempio 16-1 Esempio di un FB

Richiamo

Per richiamare questo FB si può scegliere una delle varianti seguenti. Si presuppone che VARIABILE1 sia stata definita come variabile REAL nel blocco richiamante.

```
      // Richiamo assoluto, istanza globale:
FB17.DB10 (VALORE FINALE:=10, IQ1:= VARIABILE1);

      // Richiamo simbolico, istanza globale:
TEST.TEST_1 (VALORE FINALE:= 10, IQ1:= VARIABILE1) ;
```

Esempio 16-2 Esempio di richiamo di FB con blocco dati di istanza

Risultato

Dopo l'esecuzione del blocco, in VARIABILE1 è disponibile il valore calcolato per il parametro di transito IQ1.

Lettura del valore d'uscita

I due esempi sottostanti servono ad illustrare le alternative possibili per leggere il parametro d'uscita:

```
//L'accesso al parametro d'uscita
//avviene mediante
RISULTATO:= DB10.CONTROL;
//Si può però utilizzare il parametro d'uscita
//anche con un altro
//richiamo di FB, per alimentare
//direttamente un parametro di ingresso:


FB17.DB12 (EIN_1:= DB10.CONTROL);
```

Esempio 16-3 Risultato del richiamo di FB con blocco dati di istanza

16.2.5 Esempio di richiamo di un'istanza locale

Principio

Un blocco funzionale con un semplice loop FOR potrebbe essere programmato come illustrato nell'esempio 16-1, in cui si presuppone che nella tabella dei simboli sia stato definito il simbolo `TEST` per `l'FB17`.

Richiamo

Questo FB può essere richiamto come indicato nell'esempio seguente a condizione che nel blocco richiamante `VARIABILE1` sia stata dichiarata come `REAL`:

```
// Richiamo, istanza locale:
TEST_L (VALORE FINALE:= 10, IQ1:= VARIABILE1) ;
```

Esempio 16-4 Esempio di richiamo di FB come istanza locale

Nella dichiarazione di variabili deve essere dichiarata TEST_L:

```
VAR
TEST_L : TEST;
END_VAR
```

Lettura parametro d'uscita

Il parametro d'uscita CONTROL può essere letto come segue:

```
// L'accesso al parametro d'uscita
// avviene tramite
RISULTATO:= TEST_L.CONTROL;
```

Esempio 16-5 Risultato di un richiamo di FB come istanza locale

16.3 Richiamo di funzioni

Valore di ritorno

Contrariamente ai blocchi funzionali, le funzioni forniscono come risultato un valore di ritorno. Per tale motivo, le funzioni possono essere trattate come operandi. Fa eccezione la funzione con il valore di ritorno del tipo `VOID`.

P. es. nella seguente assegnazione di valore la funzione DISTANZA viene richiamata con determinati parametri:

```
LUNGHEZZA:= DISTANZA (X1:=-3, Y1:=2);
Il valore di ritorno è DISTANZA!
```

La funzione calcola il valore di ritorno che ha lo stesso nome della funzione e lo trasmette nuovamente al blocco richiamante. Qui, il valore sostituisce il richiamo della funzione.

Il valore di ritorno può essere utilizzato nei seguenti elementi di una FC o di un FB:

- in un’assegnazione di valore
- in un’espressione logica, aritmetica o di confronto
- come parametro per un ulteriore richiamo di blocco funzionale/funzione.

Fanno eccezione le funzioni di tipo VOID. Esse non possiedono alcun valore di ritorno e non possono perciò essere utilizzate in espressioni.

La figura 16-10 illustra la sintassi di un richiamo di funzione:

Richiamo di funzione

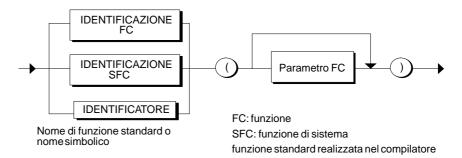



Figura 16-10 Sintassi del richiamo di una funzione

Avvertenza

Se in SCL viene richiamata una funzione il cui valore di ritorno non è stato assegnato, si può verificare una esecuzione errata del programma utente.

In una funzione SCL ciò può avvenire quando il valore di ritorno è stato assegnato ma la relativa istruzione non è stata eseguita.

In una funzione AWL/KOP/FUP ciò può avvenire se la funzione è stata programmata senza assegnazione del valore di ritorno oppure se la relativa istruzione non viene eseguita.

Richiamo

Il richiamo di una funzione avviene indicando

- il nome della funzione (IDENTIFICAZIONE FC, SFC, IDENTIFICATORE)
- e la lista dei parametri.

Esempio

Il nome della funzione che identifica il valore di ritorno può essere indicato in modo assoluto o simbolico, p. es.:

```
FC31 (X1:=5, Q1:= Somma di controllo)
DISTANZA (X1:=5, Q1:= Somma di controllo)
```

Risultato del richiamo

Dopo il richiamo, i risultati di un richiamo di funzione sono disponibili come

- valore di ritorno oppure
- come parametri di uscita o di transito (parametri attuali).

Per ulteriori informazioni al riguardo si rimanda al capitolo 18.

16.3.1 Parametri FC

Principio

Contrariamente ai blocchi funzionali, le funzioni non possiedono alcuna memoria in cui poter memorizzare i valori dei parametri. Durante l'esecuzione della funzione, i dati locali vengono memorizzati solo temporaneamente. Per questo motivo, durante il richiamo di una funzione si devono assegnare dei parametri attuali a tutti i parametri d'ingresso, di transito e d'uscita che sono stati definiti nella parte dichiarazioni di una funzione.

La figura 16-11 illustra la sintassi dell'assegnazione di parametri FC:

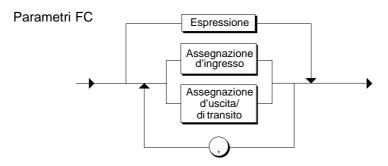



Figura 16-11 Sintassi dell'assegnazione di parametri FC

Un richiamo con assegnazioni dei parametri di ingresso, uscita e transito potrebbe avere p. es. l'aspetto seguente:

```
FC32 (E_Param1:=5,D_Param1:=LUNGHEZZA,
      A_Param1:=Somma di controllo)
```

Regole

Per l'assegnazione dei parametri valgono le regole seguenti:

- La sequenza delle assegnazioni è irrilevante.
- I tipi di dati dei parametri formali e attuali devono coincidere.
- Le singole assegnazioni sono separate fra loro mediante virgole.

16.3.2 Assegnazione d'ingresso (FC)

Principio

Mediante assegnazioni d'ingresso vengono assegnati dei parametri attuali ai parametri d'ingresso formali dell'FC richiamato. L'FC può lavorare con questi parametri, ma non può modificarli. Contrariamente al richiamo FB, nel richiamo FC questa assegnazione **non è opzionale**. Le assegnazioni d'ingresso hanno la seguente sintassi:

Sintassi

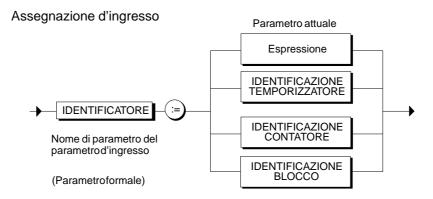



Figura 16-12 Sintassi dell'assegnazione d'ingresso

Parametri attuali nelle assegnazioni d'ingresso

Nelle assegnazioni d'uscita si possono assegnare i seguenti parametri attuali:

Tabella 16-4 Parametri attuali nelle assegnazioni d'uscita

Parametri attuali	Spiegazione
Espressione	Un'espressione rappresenta un determinato valore e si compone di operandi e operatori. Sono disponibili i seguenti tipi di espressioni: Espressione aritmetica, logica o di confronto Costante Variabile ampliata
Identificazione TEMPORIZZATORE/ CONTATORE	Definiscono un determinato temporizzatore o un determinato contatore che deve essere utilizzato per l'elaborazione di un blocco (vedere anche capitolo 17).
Identificazione BLOCCO	Definiscono un determinato blocco che deve essere utilizzato come parametro d'ingresso. Il tipo di blocco (FB, FC, DB) viene definito nella dichiarazione del parametro d'ingresso. Nell'assegnazione dei parametri si deve indicare l'indirizzo del blocco. Quest'ultimo può essere indicato in modo assoluto o simbolico (vedere anche capitolo 9).

Particolarità

Si osservi che nel caso di parametri di ingresso formali FC con tipi di dati non semplici, i parametri di transito FB e parametri FC non sono consentiti come parametri attuali.

16.3.3 Assegnazione d’uscita/di transito (FC)

Principio

In un’assegnazione d’uscita si definisce in quale variabile dell’FB richiamante devono essere scritti i valori d’uscita che vengono generati durante l’elaborazione di una funzione. Con un’assegnazione di transito si assegna un valore attuale ad un parametro di transito.

La figura 16-13 illustra la sintassi delle assegnazioni di uscita e di transito:

Assegnazione d’uscita/di transito

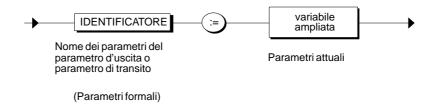



Figura 16-13 Sintassi dell’assegnazione d’uscita/di transito

Parametri attuali nelle assegnazioni d’uscita/di transito

I parametri attuali nelle assegnazioni d’uscita e di transito devono essere delle variabili poiché l’FC deve scrivere dei valori in tali parametri. Per tale motivo, i parametri d’ingresso non possono essere assegnati in assegnazioni di transito (in tal caso non sarebbe possibile scrivere il valore). Perciò, nelle assegnazioni d’uscita/di transito si può assegnare solo la variable ampliata:

Tabella 16-5 Parametri attuali nelle assegnazioni d’uscita e di transito

Parametri attuali	Spiegazione
Variabile ampliata	Sono disponibili i seguenti tipi di variabili ampliate: variabili e parametri semplici accesso a variabili assolute accesso a blocchi dati richiamo di funzioni (vedere anche capitolo 14).

Particolarità

Si osservino le seguenti particolarità:

- Dopo l'esecuzione del blocco viene attualizzato il valore modificato del parametro di transito.
- Nei parametri di transito di un tipo di dati **non semplice**, non sono consentiti come parametri attuali:
  - parametri d'ingresso di FB
  - parametri di transito di FB oppure
  - parametri di FC
- **Parametro ANY:** di regola vale anche per questo parametro quanto detto sopra. In parametri di transito di un tipo di dati **non semplici** non sono ammessi come parametri attuali:
  - parametri di ingresso di FB
  - parametri di ingresso di FC

  Inoltre, occorre considerare che le costanti **non** sono permesse come parametri attuali. Se si è dichiarato il tipo ANY come risultato della funzione (valore di ritorno) valgono anche le regole seguenti:
  - Tutti i parametri ANY devono essere provvisti di operandi il cui tipo di dati appartenga alla stessa classe. Per classe si intende p. es. la quantità dei tipi di dati numerici (INT, DINT, REAL) oppure la quantità dei tipi di dati binari (BOOL, BYTE, WORD, DWORD). I restanti tipi di dati formano una classe a sè stante.
  - Il compilatore SCL presuppone che il tipo di dati del risultato attuale della funzione sia il più importante tra i parametri attuali assegnati ai parametri ANY.
    Con il risultato della funzione sono consentite, quindi, tutte le operazioni definite per questo tipo di dati.
- **Parametro POINTER:** di regola vale anche per questo parametro quanto detto sopra. In parametri di transito di un tipo di dati **non semplici** non sono ammessi come parametri attuali:
  - parametri di ingresso di FB
  - parametri di ingresso di FC

16.3.4 Esempio di richiamo di funzioni

Principio

Una funzione DISTANZA per il calcolo della distanza di due punti (X1,Y1) e (X2,Y2) nel piano con l'impiego di un sistema di coordinate cartesiane potrebbe avere l'aspetto seguente (negli esempi riportati si presuppone sempre che il simbolo DISTANZA sia stato definito in una tabella dei simboli per FC 37).

```
FUNCTION DISTANZA: REAL
      VAR_INPUT
            X1: REAL;
            X2: REAL;
            Y1: REAL;
            Y2: REAL;
      END_VAR
      VAR_OUTPUT
            Q2: REAL;
      END_VAR
      BEGIN
            DISTANZA:= SQRT
            ( (X2-X1)**2 + (Y2-Y1)**2 );
            Q2:= X1+X2+Y1+Y2;
END_FUNCTION
```

Esempio 16-6 Calcolo della distanza

Per l'ulteriore utilizzo di un valore di una funzione sono disponibili fra l'altro le seguenti possibilità:

in un'assegnazione di valori, p. es.:

```
LUNGHEZZA:= DISTANZA (X1:=-3, Y1:=2, X2:=8.9,
Y2:=7.4, Q2:=Somma di controllo);
```

in un'espressione aritmetica o logica, p. es.:

```
RADIUS + DISTANZA (X1:=-3, Y1:=2, X2:=8.9,
Y2:=7.4, Q2:=Somma di controllo)
```

in un'assegnazione dei parametri di un blocco che viene richiamato, p. es.:

```
FB32 (DISTANZ:= DISTANZA (X1:=-3, Y1:=2, X2:=8.9,
Y2:=7.4, Q2:=Somma di controllo);
```

Esempio 16-7 Calcolo di valori all'interno di un'FC

16.4 Parametri definiti implicitamente

Panoramica

I parametri definiti implicitamente possono essere utilizzati senza doverli prima definire in un blocco. In SFC, sono disponibili i seguenti parametri definiti implicitamente:

- il parametro d'ingresso EN e
- il parametro d'uscita ENO

Entrambi i parametri sono del tipo BOOL e sono depositati nell'area dei dati di blocco temporanei.

Parametro d'ingresso EN

Ogni blocco funzionale e ogni funzione possiede il parametro d'ingresso EN definito implicitamente. Se EN è uguale a TRUE, il blocco richiamato viene eseguito, in caso contrario esso non viene eseguito. L'assegnazione del parametro EN è opzionale.

Si osservi che EN non può essere dichiarato nella parte dichiarazioni di un blocco o di una funzione.

Poiché EN è un parametro d'ingresso, EN non può essere modificato all'interno di un blocco.

Avvertenza

Il valore di ritorno di una funzione non è definito, se la funzione non è stata richiamata a causa di EN:=FALSE.

Esempio

L'esempio seguente illustra l'uso del parametro EN:

```
FUNCTION_BLOCK FB57
VAR
      RISULTATO   : REAL;
      MIA_ENABLE  : BOOL;
END_VAR
...
BEGIN
MIA_ENABLE:= FALSE;
// Richiamo di una funzione
// in cui viene assegnato il parametro EN:

RISULTATO:= FC85 (EN:= MIA_ENABLE, PAR_1:= 27);
// FC85 non è stato eseguito, perchè MIA_ENABLE
// è stato settato uguale FALSE
//....
END_FUNCTION_BLOCK
```

Esempio 16-8 Uso di EN

Parametro d'uscita ENO

Ogni blocco funzionale e ogni funzione possiede il parametro d'uscita ENO definito implicitamente, il quale è un parametro del tipo BOOL. Alla fine dell'esecuzione di un blocco, il valore attuale della variabile OK (OK Flag) viene scritto in ENO.

Subito dopo il richiamo di un blocco, in base al valore di ENO, si può verificare se tutte le operazioni del blocco sono state eseguite correttamente o se si sono verificati degli errori.

Esempio

L'esempio seguente illustra l'uso del parametro EN0.

```
FUNCTION_BLOCK FB57
      //...
      //...
BEGIN
// Richiamo di un blocco funzionale:
FB30.DB30 (X1:=10, X2:=10.5);

// Verificare se nel blocco richiamato
// tutto si è svolto correttamente:

IF ENO THEN
      // tutto corretto
      //...
ELSE
      // verificatisi errori,
      // per cui attivare il rimedio errori
      //...
END_IF;
      //...
      //...
END_FUNCTION_BLOCK
```

Esempio 16-9 Uso di ENO

Esempio

L'esempio seguente illustra una combinazione di EN e EN0.

```
// EN e ENO possono anche essere combinati fra loro
//:

FB30.DB30(X1:=10, X2:=10.5);

// La funzione seguente deve essere eseguita
// solo se l'FB 30 è stato eseguito
// senza errori:

RISULTATO:= FC 85 (EN:= ENO, PAR_1:= 27);
```

Esempio 16-10 Uso di EN e ENO

Contatori e temporizzatori

17

Panoramica

In SCL, l’esecuzione del programma può essere comandata in base all’indicazione di un temporizzatore o allo stato di un contatore.

A tal fine, STEP 7 mette a disposizione dell’utente funzioni di temporizzazione e conteggio che non devono essere definiti prima di utilizzare il programma SCL.

Sommario del capitolo

17.1 Funzioni di conteggio

Panoramica

STEP 7 mette a disposizione una serie di funzioni standard di conteggio. Questi contatori possono essere utilizzati nel programma SCL senza doverli prima definire. Essi devono solo essere alimentati con i necessari parametri. STEP 7 offre le seguenti funzioni di conteggio:

- Contatori di conteggio in avanti (Counter Up)
- Contatori di conteggio all'indietro (Counter Down)
- Contatori di conteggio in avanti e all'indietro (Counter Up Down)

Richiamo

Le funzioni di conteggio vengono richiamate in modo analogo alle funzioni. L'identificazione di funzione può essere impiegata ovunque al posto di un operando in un'espressione a condizione che il tipo di risultato della funzione sia compatibile con il tipo dell'operando sostituito.

Tabella 17-1 Nomi delle funzioni di conteggio

Nome della funzione	Significato
S_CU	Contatore in avanti (Counter Up)
S_CD	Contatore all'indietro (Counter Down)
S_CUD	Contatore in avanti e all'indietro (Counter Up Down)

Valore della funzione

Il valore della funzione (valore di ritorno) che viene ritorato al punto di richiamo è il valore di conteggio attuale `(formato BCD)` del tipo `WORD.` Per ulteriori informazioni al riguardo si rimanda al capitolo 17.1.1.

Parametri di richiamo

La tabella 17-2 contiene i parametri di richiamo con la loro identificazione ed il loro significato per tutte e 3 le funzioni di conteggio. In generale si distinguono i seguenti tipi di parametri:

- Parametri di controllo (p. es. impostare, resettare, indicare il senso del conteggio)
- Valore di preimpostazione per uno stato del contatore.
- Uscita di stato (indica se è stato raggiunto uno stato finale del contatore).
- Stato del contatore in forma binaria

Tabella 17-2 Parametro di richiamo

Parametro	Tipo di param.	Tipo di dati	Descrizione
C_NO		COUNTER	Numero di identificazione del contatore (IDENTIFICAZIONE CONTATORE); l'area dipende dalla CPU.
CU	Ingresso	BOOL	Ingresso CU: conteggio in avanti
CD	Ingresso	BOOL	Ingresso CD: conteggio all'indietro
S	Ingresso	BOOL	Ingresso per impostazione del contatore
PV	Ingresso	WORD	Valore compreso nel campo 0 – 999 per impostare il contatore (introdotto come valore 16#<valore>, laddove il valore è in formato binario BCD
R	Ingresso	BOOL	Ingresso di resettaggio
Q	Uscita	BOOL	Stato del contatore
CV	Uscita	WORD	Stato del contatore (binario)

Esempio

In seguito al richiamo menzionato nell'esempio 17-1 durante il calcolo della funzione viene riservata un'area di memoria globale del tipo COUNTER con il nome Z12.

```
Valore di conteggio:= S_CUD (C_NO :=Z12,
                            CD    :=E.0,
                            CU    :=E.1,
                            S     :=E.2 & E.3,
                            PV    :=120,
                            R     :=FALSE,
                            CV    :=binVal,
                            Q     :=actFlag);
```

Esempio 17-1 Richiamo di una funzione di conteggio all'indietro

Richiamo dinamico

Al posto del numero di conteggio assoluto (p. es. `C_NO=Z10`), nel richiamo si può indicare anche una variabile con il tipo di dati COUNTER. Ciò presenta il vantaggio di poter eseguire un richiamo dinamico del contatore assegnando a questa variabile un altro numero assoluto ad ogni richiamo.

```
Esempio:
Function_Block CONTATORE;
Var_Input
 Miocontatore: Counter;
End_Var
:
currVAL:=S_CD (C_NO:= Miocontatore,........);
```

Regole

Poiché i valori dei parametri (p. es. CD:=E.0) sono memorizzati in modo globale, in determinati casi la loro indicazione è opzionale. Queste regole di carattere generale devono essere rispettate in fase di assegnazione dei parametri:

- Durante il richiamo si deve sempre assegnare il parametro per l'identificazione del contatore C_NO.
- A seconda della funzione del contatore si deve assegnare il parametro CU (Contatore in avanti) o il parametro CD (Contatore all'indietro).
- L'indicazione dei parametri PV (valore di preimpostazione) e S (impostazione) può essere effettuata a coppie.
- Il valore del risultato in formato BCD è sempre il valore della funzione.

Avvertenza

I nomi delle funzioni e dei parametri sono uguali nei mnemonici SIMATIC e IEC. Solo l'identificazione del contatore dipende dal mnemonico: *SIMATIC:* Z e *IEC:* C

Esempio di richiamo di funzioni di conteggio

L’esempio 17-2 illustra il richiamo delle funzioni di conteggio:

```
Function_block FB1

VAR
 currVal, binVal: word;
 actFlag: bool;
END_VAR

BEGIN
currVal  :=S_CD(C_NO:=Z10, CD:=TRUE, S:=TRUE,
               PV:=100, R:=FALSE, CV:=binVal,
               Q:=actFlag);

currVal  :=S_CU(C_NO:=Z11, CU:=M0.0, S:=M0,1,
               PV:=16#110, R:=M0.2, CV:=binVal,
               Q:=actFlag);

currVal  :=S_CUD(C_NO:=Z12, CD:=E.0,
               CU:=E.1,S:=E.2 & E.3, PV:=120,
               R:=FALSE,CV:=binVal, Q:=actFlag);

currVal  :=S_CD(C_NO:=Z10,CD:=FALSE,
               S:=FALSE,
               PV:=100, R:=TRUE, CV:=bVal,
               Q:=actFlag);

end_function_block
```

Esempio 17-2 Richiamo di funzioni di conteggio

17.1.1 Introduzione ed analisi del valore di conteggio

Panoramica

Per l'introduzione del valore di preimpostazione o per l'analisi dei risultati delle funzioni è necessaria la rappresentazione interna del valore di conteggio (vedere figura 17-1).

Quando il contatore viene impostato (parametro S), il valore definito dall'utente viene trascritto nel contatore. Il campo di valori è compreso fra 0 e 999. Nell'ambito di tale campo di valori si può modificare il valore del contatore utilizzando le operazioni di conteggio in avanti/all'indietro , conteggio in avanti e conteggio all'indietro.

Formato

La figura 17-1 illustra la configurazione di bit del valore di conteggio:

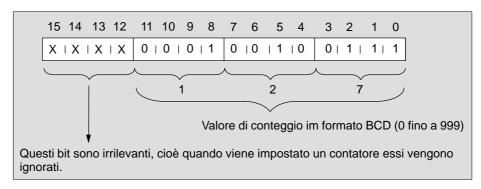



Figura 17-1 Configurazione di bit per il valore di conteggio

Introduzione

Con i formati seguenti si può caricare un valore di conteggio predefinito:

- Decimale come valore di un numero intero: p. es. 295, purché questo valore corrisponda ad un valido formato BCD.
- In formato BCD (introduzione come costante esadecimale): p. es. 16#127

Analisi

Il risultato può essere analizzato in due formati diversi:

- Come risultato della funzione (tipo WORD): in formato BCD
- Come parametro d'uscita CV (tipo WORD): binario

17.1.2 Conteggio in avanti (Counter Up)

Descrizione

Con il contatore Counter Up si possono eseguire solo operazioni di conteggio in avanti.

Modo di funzionamento

Tabella 17-3 Modo di funzionamento Contatore in avanti

Operazione	Modo di funzionamento
Conteggio in avanti	Il valore del contatore viene aumentato di ”1”, quando lo stato del segnale all’ingresso CU passa da ”0” a ”1” e il valore di conteggio è minore di 999.
Imposta contatore	Quando lo stato del segnale all’ingresso **S** passa da ”0” a ”1”, il contatore viene impostato con il valore dell’ingresso **PV**. Un tale cambio di segnale è sempre necessario per poter impostare un contatore.
Resetta	Il contatore viene resettato quando l’ingresso **R** = 1. Il reset del contatore imposta il valore di conteggio a ”0”.
Interrogazione contatore	Un’interrogazione dello stato del segnale all’uscita Q risulta ”1” se il valore di conteggio è maggiore di ”0”. L’interrogazione risulta ”0” se il valore di conteggio è uguale a ”0”.

17.1.3 Conteggio all’indietro (Counter Down)

Descrizione

Con il contatore Counter Down si possono eseguire solo operazioni di conteggio all’indietro.

Modo di funzionamento

Tabella 17-4 Modo di funzionamento Contatore all’indietro

Operazione	Modo di funzionamento
Conteggio all’indietro	Il valore del contatore viene diminuito di ”1”, quando lo stato del segnale all’ingresso **CD** passa da ”0” a ”1” e il valore di conteggio è maggiore di 999.
Imposta contatore	Quando lo stato del segnale all’ingresso **S** passa da ”0” a ”1”, il contatore viene impostato con il valore dell’ingresso **PV**. Un tale cambio di segnale è sempre necessario per poter impostare un contatore.
Resetta	Il contatore viene resettato quando l’ingresso **R** = 1. Il reset del contatore imposta il valore di conteggio a ”0”.
Interrogazione contatore	Un’interrogazione dello stato del segnale all’uscita **Q** risulta ”1” se il valore di conteggio è maggiore di ”0”. L’interrogazione risulta ”0” se il valore di conteggio è uguale a ”0”.

17.1.4 Conteggio in avanti e all’indietro (Counter Up Down)

Descrizione

Con il contatore Counter Up Down si possono eseguire operazioni di conteggio sia in avanti che all’indietro. In caso di simultaneità degli impulsi di conteggio in avanti e all’indietro, vengono eseguite entrambe le operazioni.
Il valore di conteggio rimane immutato.

Modo di funzionamento

Tabella 17-5 Modo di funzionamento Contatore in avanti e all’indietro

Operazione	Modo di funzionamento
Conteggio in avanti	Il valore del contatore viene aumentato di ”1”, quando lo stato del segnale all’ingresso **CU** passa da ”0” a ”1” e il valore di conteggio è minore di 999.
Conteggio all’indietro	Il valore del contatore viene diminuito di ”1”, quando lo stato del segnale all’ingresso **CD** passa da ”0” a ”1” e il valore di conteggio è maggiore di 999.
Imposta contatore	Quando lo stato del segnale all’ingresso **S** passa da ”0” a ”1”, il contatore viene impostato con il valore dell’ingresso **PV**. Un tale cambio di segnale è sempre necessario per poter impostare un contatore.
Resetta	Il contatore viene resettato quando l’ingresso **R** = 1. Il reset del contatore imposta il valore di conteggio a ”0”.
Interrogazione contatore	Un’interrogazione dello stato del segnale all’uscita Q risulta ”1” se il valore di conteggio è maggiore di ”0”. L’interrogazione risulta ”0” se il valore di conteggio è uguale a ”0”.

17.1.5 Esempio della funzione S_CD (Contatore all’indietro)

Occupazione dei parametri

La tabella 17-6 illustra l’occupazione dei parametri della funzione riportata nell’esempio S_CD.

Tabella 17-6 Parametro di richiamo

Parametro	Descrizione
C_NO	MIOCONTATORE
CD	Ingresso E0.0
S	SETTARE
PV	Preimpostazione 16#0089
R	Resettaggio
Q	A0.7
CV	BIN_WERT

Esempio

L'esempio 17-3 illustra la funzione di conteggio S_CD:

```
FUNCTION_BLOCK CONTARE
VAR_INPUT
 IOCONTATORE: COUNTER;
END_VAR
VAR_OUTPUT
 RISULTATO: INT;
END_VAR
VAR
 IMPOSTARE        : BOOL;
 RESETTARE        : BOOL;
 VALORE_BCD       : WORD; //Stato contatore
                          //codice BCD
 VALORE_BCD       : WORD; //Stato cont. binario
 PREIMPOSTAZIONE  : WORD;
END_VAR
BEGIN
 A0.0:= 1;
 IMPOSTARE:= E0.2;
 RESETTARE:= E0.3;
 PREIMPOSTAZIONE:= 16#0089;
 VALORE_BCD:= S_CD
            (C_NO := MIOCONTATORE,//Conteggio
                   in avanti
            CD    := E.0,
            S     := IMPOSTARE ,
            PV    := PREIMPOSTAZIONE,
            R     := RESETTARE,
            CV    := VALORE_BIN,
            Q     := A0.7);
RISULTATO:=WORD_TO_INT(VAL_BIN);//ULTER. ELA.
                  //come parametro d'uscita
AW4:= VALORE_BCD; //All'uscita per visualizzazione
END_FUNCTION_BLOCK
```

Esempio 17-3 Esempio di una funzione di conteggio

17.2 Funzioni di temporizzazione (TIMER)

Panoramica

I temporizzatori sono elementi funzionali nel programma utente, che eseguono e sorvegliano eventi comandati a tempo. STEP 7 mette a disposizione dell'utente una serie di funzioni di temporizzazione standard alle quali si può accedere tramite SCL. Con le operazioni di temporizzazione nel programma utente si può:

- regolare tempi di attesa
- consentire tempi di monitoraggio
- generare impulsi
- misurare tempi

Richiamo

Le funzioni di temporizzazione vengono richiamate in modo analogo a quello delle funzioni di conteggio. L'identificazione della funzione può essere inserita ovunque in un'espressione, al posto di un operando, purché il tipo dei risultati della funzione sia compatibile con il tipo del primo operando.

Tabella 17-7 Funzioni di temporizzazione STEP 7

Nome della funzione	Significato
S_PULSE	impulso (Pulse)
S_PEXT	impulso prolungato (Pulse Extended)
S_ODT	ritardo all'insezione (On Delay Time)
S_ODTS	ritardo all'insezione con memoria (Stored On Delay Time)
S_OFFDT	ritardo alla disinserzione (Off Delay Time)

Valore della funzione

Il valore della funzione (valori di ritorno) che viene ritornato al punto di richiamo è un valore temporale con tipo di dati S5TIME. Per informazioni al riguardo si rimanda al capitolo 17.2.1.

Parametri di richiamo

I parametri da assegnare vengono spiegati in modo tabellare nella descrizione delle rispettive funzioni standard. I nomi con i corrispondenti tipi di dati per tutte e 5 le funzioni di temporizzazione sono contenuti nella tabella 17-8.

In generale, si distinguno i seguenti tipi di parametri:

- Parametri di controllo (p. es. impostazione, resettaggio)
- Valore di preimpostazione per il tempo di avvio.
- Uscita di stato (indica se il temporizzatore è ancora in funzione)
- Valore di tempo residuo in forma binaria.

Tabella 17-8 Parametri di richiamo

Parametro	Tipo di dati	Descrizione
T_NO	TIMER	Identificazione di tempo; l’area dipende dalla CPU
S	BOOL	Ingresso di avvio
TV	S5TIME	Preimpostazione valore di tempo (formato BCD)
R	BOOL	Ingresso di resettaggio
Q	BOOL	Stato del temporizzatore
BI	WORD	Valore di tempo residuo (binario)

Esempio

Al termine dell’esecuzione del richiamo illustrato nell’esempio 17-4, un’area di dati globale del tipo TIMER viene riservata con il nome T10.

```
RITARDO:=     S_ODT (T_NO :=    T10,
                     S    :=    TRUE,
                     TV   :=    T#1s,
                     R    :=    FALSE,
                     BI   :=    biVal,
                     Q    :=    actFlag
                     );
```

Esempio 17-4 Richiamo di una funzione di conteggio all’indietro

Richiamo dinamico

Durante il richiamo, al posto del numero di temporizzatore assoluto (p. es. T10), si può anche indicare una variabile di tipo TIMER. Ciò presenta il vantaggio di poter organizzare in modo dinamico il richiamo del temporizzatore assegnando a questa variabile un numero assoluto diverso ad ogni richiamo.

```
Esempio:
FUNCTION_BLOCK TIMER
VAR_INPUT
miotemporizzatore: timer;
END_VAR
:
currTime:=S_ODT (T_NO:=miotemporizzatore,.........)
```

Regole

Poiché i valore dei parametri sono memorizzati in modo globale, in determinati casi la loro indicazione è opzionale. Per l'assegnazione di parametri si devono osservare le seguenti regole generali:

- il parametro per l'identificazione del temporizzatore T_NO deve essere indicato, al suo richiamo, in forma simbolica o assoluta.
- L'indicazione del parametro PV (valore di preimpostazione) e S (impostazione) può essere omessa a coppie.
- L'assegnazione dei parametri è opzionale. Si può accedere a Q e BI mediante un'assegnazione di valori.
- Il valore del risultato in formato S5TIME è sempre il valore della funzione.

Avvertenza

I nomi delle funzioni sono identici sia nel mnemonico SIMATIC che IEC.

Esempio: richiamo di funzioni di temporizzazione

L'esempio 17-5 illustra il richiamo delle funzioni di temporizzazione:

```
FUNCTION_BLOCK FB2

VAR
currTime: S5time;
biVal: word;
actFlag: bool;
END VAR

BEGIN
currTime:= S_ODT (T_NO:=T10, S:=TRUE, TV:=T#1s,
                  R:=FALSE, BI:=biVal,
                  Q:=actFlag);

currTime:= S_ODTS (T_NO:=T11, S:=M0,0, TV:=T#1s,
                  R:= M0.1, BI:=biVal,
                  Q:= actFlag);

currTime:=S_OFFDT (T_NO:=T12, S:=I0.1&actFlag,
                  TV:= T#1s,R:=FALSE,BI:=biVal,
                  Q:= actFlag);

currTime:= S_PEXT (T_NO:=T13, S:=TRUE,
                  TV:=T#1s,R:=I0.0, BI:=biVal,
                  Q:=actFlag);

currTime:= S_PULSE (T_NO:=T14, S:=TRUE,
                   TV:=T#1s,R:=FALSE, BI:=biVal,
                    Q:=actFlag);
END_FUNCTION_BLOCK
```

Esempio 17-5 Richiamo di funzioni di temporizzazione

17.2.1 Introduzione ed analisi del valore di conteggio

Panoramica

Per l'introduzione del valore di preimpostazione o per l'analisi dei risultati della funzione in codice BCD è necessaria la rappresentazione interna del valore di tempo (vedere figura 17-2).

Ogni aggiornamento del tempo riduce il valore di tempo di un'unità in un intervallo che dipende dalla base del tempo. Il valore di tempo viene ridotto finché esso non sia uguale a "0". Il campo di temporizzazione va da 0 fino a 9 990 secondi.

Formato

La figura 17-2 illustra la rappresentazione interna del valore di tempo.

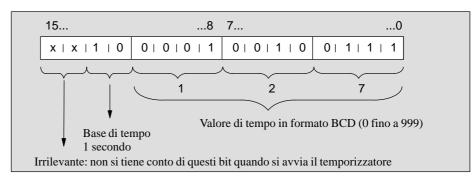



Figura 17-2 Rappresentazione del valore di tempo

Introduzione

Con i seguenti formati si può caricare un valore di tempo predefinito:

- in rappresentazione a livelli: `TIME#aH_bbM_ccS_dddMS`
- in rappresentazione decimale: `TIME#2.4H`

Analisi

Il risultato può essere analizzato in due formati diversi:

- come risultato della funzione (tipo S5TIME): in formato BCD
- come parametro d'uscita (valore di tempo senza base di tempo del tipo WORD): binario.

Base di tempo

I bit 12 e 13 della parola di tempo contengono la base di tempo in codice binario. La base di tempo definisce l'intervallo in cui il valore di tempo viene ridotto di un'unità (vedere tabella 17-9 e figura 17-2). La minima base di tempo è 10 ms; la massima è 10 s.

Tabella 17-9 Base di tempo e codice binario

Base di tempo	Codice binario per base di tempo
10 ms	00
100 ms	01
1 s	10
10 s	11

Avvertenza

Poiché i valori di tempo vengono memorizzati solo in un determinato intervallo di tempo, i valori che non corrispondono ad un multiplo esatto dell'intervallo di tempo vengono tagliati.

I valori, la cui risoluzione è troppo alta per l'area desiderata, vengono arrotondati in modo da raggiungere l'area desiderata, ma **non** la risoluzione desiderata.

17.2.2 Avviamento del temporizzatore come impulso

Descrizione

Il tempo massimo in cui il segnale d'uscita rimane ad ”1” equivale al valore del tempo programmato t.

Se durante il tempo di esecuzione del temporizzatore sull'ingresso si verifica lo stato di segnale 0, il temporizzatore viene impostato ad ”0”. Ciò significa che il tempo di esecuzione termina prima del tempo previsto.

La figura 17-3 illustra il modo di funzionamento del temporizzatore ”Avvia temporizzatore come impulso”:

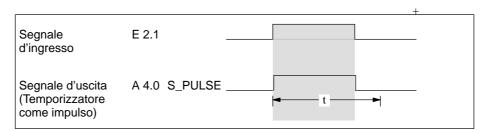



Figura 17-3 Temporizzatore ”Avvia temporizzatore come impulso”

Modo di funzionamento

Tabella 17-10 Modo di funzionamento ”Avvia temporizzatore come impulso”

Operazione	Modo di funzionamento
Avvia temporizzatore	L'operazione ”Avvia temporizzatore come impulso” avvia un temporizzatore indicato quando lo stato del segnale all'ingresso di avvio (**S**) passa da ”0” a ”1”. Per abilitare il temporizzatore è sempre necessario un cambio del segnale.
Definisci il tempo di esecuzione	Il temporizzatore continua a scorrere con il valore indicato all'ingresso **TV**, fino allo scadere del tempo programmato e l'ingresso S = 1.
Fine del tempo di esecuzione prima del previsto	Se l'ingresso S passa da ”1” a ”0” prima che il valore del tempo sia scaduto, il temporizzatore viene interrotto.
Resetta	Il temporizzatore viene resettato quando l'ingresso di resettaggio (**R**) passa da ”0” a 1”, durante lo scorrimento del tempo. In seguito a questo cambio, anche il valore del tempo e la base di tempo vengono resettate a zero. Lo stato del segnale ”1” all'ingresso R non è rilevante se il temporizzatore non è in funzione.
Interroga stato del segnale	Finché il temporizzatore è in funzione, un'interrogazione dello stato del segnale ”1” all'ingresso **Q** fornisce il risultato ”1”. In caso di fine del tempo di esecuzione prima del tempo previsto, un'interrogazione dello stato del segnale all'ingresso Q fornisce il risultato ”0”.
Interroga valore del tempo attuale	Il valore del tempo attuale può essere interrogato all'uscita **BI** tramite il valore della funzione S_PULSE.

17.2.3 Avviamento del temporizzatore come impulso prolungato

Descrizione

Il segnale d'uscita per il tempo programmato (t) rimane ad "1" indipendentemente dal tempo in cui il segnale d'ingresso rimane a "1". Una nuova risoluzione dell'impulso di avvio causando un nuovo scorrimento del tempo, ritarda anche l'impulso d'uscita (post-attivazione).

La figura 17-4 illustra die modo di funzionamento del temporizzatore "Avvia temporizzatore come impulso prolungato".

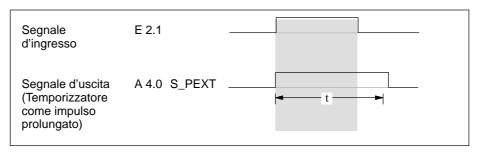



Figura 17-4 Temporizzatore "Avvia temporizzatore come impulso prolungato"

Tabella 17-11 Modo di funzionamento "Avvia temporizzatore come impulso prolungato"

Modo di funzionamento

Operazione	Modo di funzionamento
Avvia temporizzatore	L'operazione "Avvia temporizzatore come impulso prolungato" avvia un tempo indicato quando lo stato del segnale all'ingresso di avvio (**S**) passa da "0" a "1". Per abilitare il temporizzatore è sempre necessario un cambio di segnale.
Avvia nuovamente il tempo di esecuzione	Se lo stato del segnale all'ingresso **S** durante il tempo di esecuzione passa nuovamente a "1", il tempo viene avviato nuovamente con il valore del tempo indicato.
Preimposta tempo di esecuzione	Il temporizzatore continua a funzionare con il valore indicato all'ingresso **TV**, fino allo scadere del tempo programmato.
Resetta	Il temporizzatore viene resettato quando l'ingresso di reset (**R**) passa da "0" a 1", durante lo scorrimento del temporizzatore. In seguito a questo cambio, anche il valore del tempo e la base di temporizzazione vengono resettate a zero. Lo stato del segnale "1" all'ingresso R non è rilevante se il temporizzatore non è in funzione.
Interroga stato del segnale	Finché il temporizzatore è in funzione, un'interrogazione dello stato del segnale "1" all'ingresso **Q** fornisce il risultato "1", indipendentemente dalla lunghezza del segnale d'ingresso.
Interroga valore attuale del tempo	Il valore di tempo attuale può essere interrogato all'uscita **BI** tramite il valore della funzione S_PEXT.

17.2.4 Avviamento del temporizzatore come ritardo all’insezione

Descrizione

Il segnale d’uscita passa da ”0” a ”1” quando il tempo programmato è scaduto e il segnale d’ingresso è ancora ”1”, vale a dire che l’uscita viene inserita con ritardo. I segnali d’ingresso la cui durata del tempo è più breve del programmato, non appaiono all’uscita.

La figura 17-5 illustra il modo di funzionamento del temporizzatore ”Avvia temporizzatore come ritardo all’inserzione”.

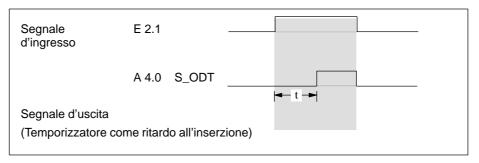



Figura 17-5 Temporizzatore ”Avvia temporizzatore come ritardo all’inserzione”

Modo di funzionamento

Tabella 17-12 Modo di funzionamento ” Avvia temporizzatore come ritardo all’inserzione”

Operazione	Modo di funzionamento
Avvia temporizzatore	L’operazione ”Avviare il temporizzatore come ritardo all’inserzione” avvia un temporizzatore indicato quando lo stato del segnale all’ingresso di avvio (**S**) passa da ”0” a ”1”. Per abilitare il temporizzatore è sempre necessario un cambio del segnale.
Interrompi temporizzatore	Il temporizzatore viene interrotto, durante il suo funzionamento, se lo stato del segnale all’ingresso **S** passa da ”1” a ”0”.
Definisci tempo di esecuzione	Il temporizzatore continua a scorrere con il valore indicato all’ingresso **TV** finché lo stato del segnale all’ingresso S = 1.
Resetta	Il temporizzatore viene resettato quando l’ingresso di resettaggio **R** passa da ”0” a 1”, durante lo scorrimento del temporizzatore. In seguito a questo cambio, anche il valore del tempo e la base di tempo vengono resettati a zero. Lo stato del segnale ”1” all’ingresso R non è rilevante se il temporizzatore non è in funzione.
Interroga stato del segnale	Un’interrogazione dello stato del segnale per ”1” all’uscita **Q** risulta ”1” se il tempo è scaduto senza errori e l’ingresso **S** è ancora ”1”. Se il temporizzatore è stato interrotto, un’interrogazione dello stato del segnale ”1” risulta sempre ”0”. Un’interrogazione dello stato del segnale per ”1” all’uscita Q risulta sempre ”0” anche se il temporizzatore non è in funzione e l’RLC all’ingresso S è sempre ”1”.
Interroga valore attuale di tempo	Il valore di tempo attuale può essere interrogato all’uscita **BI** tramite il valore della funzione S_ODT.

17.2.5 Avviamento del temporizzatore come ritardo all'inserzione con memoria

Descrizione

Il segnale d'uscita passa da ”0” a ”1” allo scadere del tempo programmato, indipendentemente dal tempo in cui il segnale d'ingresso rimane ad ”1”.

La figura 17-6 illustra il modo di funzionamento del temporizzatore ”Avvia temporizzatore come ritardo all'inserzione con memoria”.

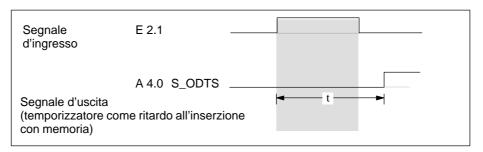



Figura 17-6 Temporizzatore ”Avvia temporizzatore come ritardo all'inserzione con memoria”

Tabella 17-13 Modo di funzionamento ”Avvia temporizzatore come ritardo all'inserzione con memoria”

Modo di funzionamento

Operazione	Modo di funzionamento
Avvia temporizzatore	L'operazione ”Avvia temporizzatore come ritardo alla disinserzione con memoria” avvia un temporizzatore indicato quando lo stato del segnale all'ingresso di avvio **S** passa da ”0” a ”1”. Per abilitare il temporizzatore è sempre necessario un cambio del segnale.
Riavvia temporizzatore	Il temporizzatore viene riavviato con il valore indicato, quando lo stato del segnale all'ingresso **S** passa nuovamente da ”0” a ”1”, mentre il tempo scorre.
Definisci tempo di esecuzione	Il tempo continua a scorrere con il valore indicato all'ingresso **TV** anche quando lo stato del segnale all'ingresso **S** passa a ”0” ancora prima dello scadere del tempo.
Resetta	Se l'ingresso di reset **R** passa da ”0” a ”1”, il tempo viene resettato all'ingresso S indipendentemente dall'RLC (per il risultato della combinazione vedere **/232/**).
Interroga stato del segnale	Un'interrogazione dello stato di segnale per ”1” all'uscita **Q** risulta ”1”, indipendentemente dallo stato del segnale all'ingresso S.
Interroga valore attuale del tempo	Il valore del tempo attuale può essere interrogato all'uscita **BI** tramite il valore della funzione S_ ODTS.

17.2.6 Avviamento del temporizzatore come ritardo alla disinserzione

Descrizione

In caso di passaggio dello stato del segnale da ”0” a ”1” all’ingresso S, all’uscita Q appare lo stato ”1”. Se all’ingresso di avvio lo stato passa da ”1” a ”0”, il temporizzatore viene attivato. L’uscita del segnale assume lo stato ”0” solo allo scadere del tempo. L’uscita viene avviata con ritardo all’inserzione.

La figura 17-7 illustra il modo di funzionamento del temporizzatore ”Avvia temporizzatore come ritardo alla disinserzione”.

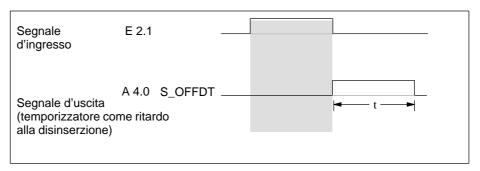



Figura 17-7 Temporizzatore ”Avvia temporizzatore come ritardo alla disinserzione”

Modo di funzionamento

Tabella 17-14 Modo di funzionamento ” Avvia temporizzatore come ritardo alla disinserzione”

Operazione	Modo di funzionamento
Avvia temporizzatore	L’operazione Avvia temporizzatore come ritardo alla disinserzione avvia il temporizzatore indicato quando lo stato del segnale **S** passa da ”1” a ”0”. Per abilitare il temporizzatore è necessario sempre un cambio del segnale.
Riavvia temporizzzatore	Il temporizzatore viene riavviato quando lo stato del segnale all’ingresso **S** passa nuovamente da ”1” a ”0” (p. es. dopo il resettaggio).
Definisci tempo di esecuzione	Il tempo scorre con il valore indicato all’ingresso **TV**.
Resetta	Il temporizzatore viene resettato se l’ingresso di resettaggio (**R**) passa da ”0” a ”1”, mentre il tempo scorre.
Interroga stato del segnale	Una interrogazione del segnale per ”1” all’uscita **Q** ha come risultato ”1” se lo stato del segnale all’ingresso S = 1 oppure se il tempo scorre.
Interroga valore di tempo attuale	Il valore di tempo attuale può essere interrogato all’uscita **BI** e con il valore della funzione S_OFFDT.

17.2.7 Esempio di programma per un impulso prolungato

Esempio S_PEXT

L'esempio 17-6 illustra un programma per l'impiego della funzione di temporizzazione "impulso prolungato".

```
FUNCTION_BLOCK SENSORETEMPORALE
VAR_INPUT
 MIOTEMPORIZZATORE: TIMER;
END_VAR
VAR_OUTPUT
 RISULTATO: S5TIME;
END_VAR
VAR
 IMPOSTA          : BOOL;
 RESETTA          : BOOL;
 VALORE_BCD       : S5TIME;//Base di tempo e
                   //val. res. in codice BCD
 VALORE_BIN       : WORD; //Val. di tempo bin.
 PREIMPOSTAZIONE  : S5TIME;
END_VAR
BEGIN
 A0.0:= 1;
 IMPOSTA:= E0.0;
 RESETTA:= E0.1;
 PREIMPOSTAZIONE:= T#25S;

 VALORE_BCD:= S_PEXT(T_NO:= MIO TEMPORIZZATORE,
                  S   := IMPOSTA,
                  TV  := PREIMPOSTAZIONE,
                  R   := RESETTA,
                  BI  := VALORE BINARIO,
                  Q   := A0.7);

RISULTATO:=VALORE_BCD;//Ulteriore elaborazione
                   //come parametro d'uscita
AW4:= VALORE_BIN  //All'uscita per visualiz.
END_FUNCTION_BLOCK
```

Esempio 17-6 Esempio di funzione di temporizzazione

17.2.8 Scelta del giusto temporizzatore

La figura 17-8 offre una panoramica dei cinque temporizzatori descritti in questo paragrafo. Questa panoramica intende essere un aiuto per facilitare la scelta dei temporizzatori più adatti alle esigenze dell'utente.

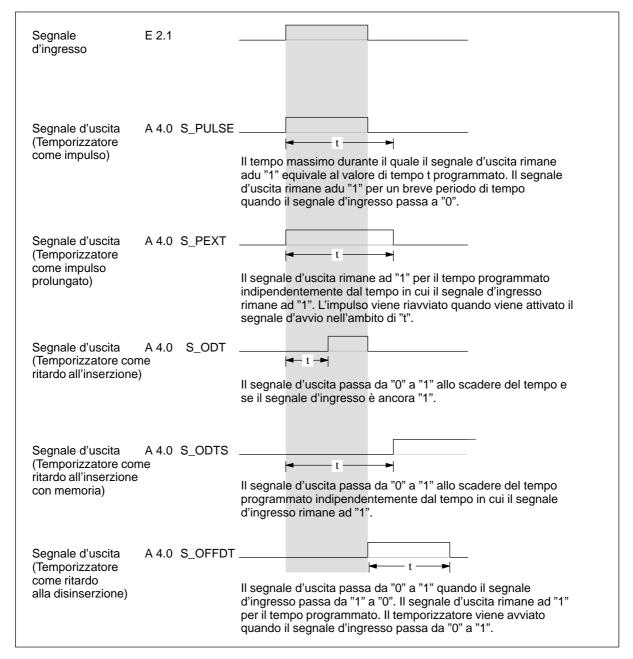



Figura 17-8 Scelta del giusto temporizzatore

Funzioni standard SCL

18

Panoramica

Per risolvere i problemi che si verificano più frequentemente, SCL mette e disposizione dell'utente una serie di funzioni standard che possono essere richiamatie nei propri blocchi SCL.

Sommario del capitolo

18.1 Conversione dei tipi di dati

Panoramica

Quando si combinano fra loro due operandi con tipi di dati diversi o si assegnano espressioni a delle variabili, nei singoli casi occorre tener conto della compatibilità dei tipi di dati. Nei casi seguenti si ottengono risultati errati:

- quando si passa ad un'altra classe, p. es. da un tipo di dati bit a un tipo di dati numerico
- nell'ambito di una classe, quando il tipo di dati di destinazione è meno importante del tipo di dati sorgente.

Perciò, in tali casi si deve eseguire una conversione **esplicita** dei tipi di dati. Per informazioni al riguardo si rimanda al capitolo 18.2.

Se non si verificano i due casi suddetti, il compilatore esegue una conversione implicita dei tipi di dati. Di conseguenza tale operazione viene denominata conversione **implicita** dei tipi di dati.

Conversione implicita dei tipi di dati

Nell'ambito delle classi di tipi di dati ausiliari, definite nella tabella 18-1, il compilatore esegue una conversione implicita dei tipi di dati nella sequenza indicata. Come formato comune di due operandi viene definito il tipo di dati più piccolo il cui campo di valori contiene entrambi gli operandi – p. es. il formato comune di Byte e Integer – Integer.

Si prega di osservare che in una conversione di tipi di dati nell'ambito della classe ANY_BIT i bit iniziali vengono impostati a 0.

Tabella 18-1 Sequenza della conversione implicita dei tipi di dati

Classi	Sequenza di conversione
ANY_BIT	BOOL ⇒ BYTE ⇒ WORD ⇒ DWORD
ANY_NUM	INT ⇒ DINT ⇒ REAL

L'esempio seguente descrive la conversione implicita dei tipi di dati:

```
VAR
      PID_REGOLATORE_1 : BYTE ;
      PID_REGOLATORE_2 : WORD ;
END_VAR

BEGIN
IF (PID_REGOLATORE_1 <> PID_REGOLATORE_2) THEN ...

(* Nella condizione della suddetta istruzione
IF / THEN, PID_REGOLATORE_1 viene convertita
implicitamente in una variabile con tipo di
dati WORD. *)
```

Esempio 18-1 Conversione implicita dei tipi di dati

18.2 Funzioni standard per la conversione dei tipi di dati

Conversione esplicita dei tipi di dati

La conversione esplicita dei tipi di dati viene eseguita con funzioni standard. Queste funzioni standard sono contenute nelle tabelle 18-2 e 18-3.

Richiamo di funzioni

Per una descrizione dettagliata sul richiamo di funzioni generiche si rimanda al capitolo 16.

- Parametro d'ingresso:

Ogni funzione per la conversione di un tipo di dati possiede esattamente un parametro d'ingresso con il nome `IN`. Poiché si tratta di una funzione con un solo parametro, si deve indicare solo il parametro attuale.

- Valore della funzione

Il valore della funzione è sempre il valore di ritorno della funzione stessa. Le due tabelle illustrano le regole in base alle quali viene convertito un tipo di dati. La tabella 18-3 indica anche se la rispettiva funzione influsice sul `OK`.

- Assegnazione di nomi

Poiché i tipi di dati del parametro d'ingresso e del valore della funzione risultano dal rispettivo nome della funzione, essi non sono elencati nelle tabelle 18-2 e 18-3: p. es. nella funzione `BOOL_TO_BYTE`, il tipo di dati del parametro d'ingresso è `BOOL` e il tipo di dati del valore della funzione è `BYTE`.

Funzioni di conversione (classe A)

La tabella 18-2 illustra le funzioni di conversione di tipi di dati della classe A. Queste funzioni vengono attivate implicitamente dal compilatore, ma possono anche essere attivate esplicitamente dall'utente. Il risultato è sempre definito.

Tabella 18-2 Funzioni di conversione dei tipi di dati della classe A

Nome della funzione	Regola di conversione
BOOL_TO_BYTE	Integrazione con zeri iniziali
BOOL_TO_DWORD	
BOOL_TO_WORD	
BYTE_TO_DWORD	
BYTE_TO_WORD	
CHAR_TO_STRING	Trasformazione in una stringa (di lunghezza 1) contenente lo stesso carattere.
DINT_TO_REAL	Trasformazione in REAL secondo la norma IEEE. Il valore può essere soggetto a modifiche a causa della diversa precisione di REAL.
INT_TO_DINT	La parola più significativa del valore della funzione viene riempita con 16#FFFF in caso di parametro d'ingresso negativo, negli altri casi viene riempita con zeri. Il valore rimane lo stesso.
INT_TO_REAL	Trasformazione in REAL secondo la norma IEEE. Il valore rimane lo stesso.
WORD_TO_DWORD	Integrazione con zeri iniziali

Funzioni di conversione (classe B)

La tabella 18-3 illustra le funzioni di conversione di tipi di dati della classe B. Queste funzioni devono essere attivate esplicitamente dall'utente. Il risultato può anche essere indefinito se la dimensione del tipo di dati di destinazione non è sufficiente.

L'utente può controllare da sé un caso del genere attivando il controllo di valori limite oppure può far eseguire il controllo dal sistema selezionando l'opzione ”OK Flag” prima della compilazione. Nei casi in cui il risultato è indefinito, il sistema imposta la variabile OK su FALSE. L'analisi è a cura dell'utente.

Tabella 18-3 Funzioni di conversione dei tipi di dati della classe B

Nome della funzione	Regola di conversione	OK
BYTE_TO_BOOL	Copiatura del bit meno significativo	S
BYTE_TO_CHAR	Immissione della stringa di bit	N
CHAR_TO_BYTE	Immissione della stringa di bit	N
CHAR_TO_INT	La stringa di bit presente nel parametro d'ingresso viene registrata nel byte meno significativo del valore della funzione. Il byte più significativo viene riempito con zeri.	N
DATE_TO_DINT	Immissione della stringa di bit	N
DINT_TO_DATE	Immissione della stringa di bit	S
DINT_TO_DWORD	Immissione della stringa di bit	N
DINT_TO_INT	Copiatura del bit per il segno. Il valore presente nel parametro d'ingresso viene interpretato nel tipo di dati INT. Se il valore è minore di -32_768 o maggiore di 32_767, la variabile OK viene impostata su FALSE.	
DINT_TO_TIME	Immissione della stringa di bit	N
DINT_TO_TOD	Immissione della stringa di bit	S
DWORD_TO_BOOL	Copiatura dei bit meno significativo	S
DWORD_TO_BYTE	Copiatura degli 8 bit di valore minimo	S
DWORD_TO_DINT	Immissione della stringa di bit	N
DWORD_TO_REAL	Immissione della stringa di bit	N
DWORD_TO_WORD	Copiatura dei 16 bit meno significativi	S
INT_TO_CHAR	Immissione della stringa di bit	S
INT_TO_WORD	Immissione della stringa di bit	N
REAL_TO_DINT	Arrotondamento del valore IEEE–REAL a DINT. Se il valore è minore di -2_147_483_648 o maggiore di 2_147_483_647, la variabile OK viene impostata su FALSE.	S
REAL_TO_DWORD	Immissione della stringa di bit	N
REAL_TO_INT	Arrotondamento del valore IEEE–REAL a INT. Se il valore è minore di -32_768 o maggiore di 2_147_483_647, la variabile OK viene impostata su FALSE.	S
STRING_TO_CHAR	Copiatura del primo carattere della stringa. Se la stringa non ha una lunghezza di 1, la variabile OK viene impostata su FALSE.	S

Tabella 18-3 Funzioni di conversione dei tipi di dati della classe B

Nome della funzione	Regola di conversione	OK
TIME_TO_DINT	Immissione della stringa di bit	N
TOD_TO_DINT	Immissione della stringa di bit	N
WORD_TO_BOOL	Copiatura del bit di valore minimo	S
WORD_TO_BYTE	Copiatura degli 8 bit meno significativi	S
WORD_TO_INT	Immissione della stringa di bit	N
WORD_TO_BLOCK_DB	La stringa di bit di WORD viene interpretata come numero di blocco dati.	N
BLOCK_DB_TO_WORD	Il numero di blocco dati viene interpretato da WORD come stringa di bit.	N

Avvertenza

Si ha inoltre la possibilità di sfruttare le **funzioni IEC** per la conversione dei tipi di dati. Per ulteriori informazioni sulle singole funzioni IEC si rimanda a **/235/**.

Esempi di conversione esplicita

Nell'esempio 18-2 è necessaria una conversione esplicita poiché il tipo di dati di destinazione è meno importante del tipo di dati sorgente.

```
FUNCTION_BLOCK FB10
VAR
      INTERRUTTORE  : INT;
      REGOLATORE    : DINT;
END_VAR

BEGIN
INTERRUTTORE := DINT_TO_INT (REGOLATORE);
(* INT è meno importante di DINT *)
// . . .
END_FUNCTION_BLOCK
```

Esempio 18-2 Il tipo di dati di destinazione non è adatto al tipo di dati sorgente

Nell’esempio 18-3 viene applicata una conversione esplicita visto che il tipo di dati REAL non è consentito per un’espressione aritmetica con operatore MOD.

```
FUNCTION_BLOCK FB20
BEGIN
VAR
valoreint:INT:=17;
CONV2 := INT;
END_VAR
CONV2 := valoreint MOD REAL_TO_INT (2.3);
(* MOD può essere impiegato solo con dati del
tipo INT o DINT.  *)
// ...
END_FUNCTION_BLOCK
```

Esempio 18-3 Conversione a causa di un tipo di dati non consentito

Esempio:
Nell’esempio 18-4 è necessaria una conversione poiché non è presente il tipo di dati corretto per un operatore logico. È consentito utilizzare l’operatore NOT solo con dati del tipo BOOL, BYTE, WORD o DWORD.

```
FUNCTION_BLOCK FB30
VAR
valoreint:INT:=17;
CONV1 :=WORD;
END_VAR
BEGIN
CONV1 := NOT INT_TO_WORD(valoreint);
(* NOT può essere impiegato solo con dati
 del tipo INT. *)
// ...
END_FUNCTION_BLOCK
```

Esempio 18-4 Conversione a causa di un tipo di dati errato

L'esempio 18-5 illustra la conversione nel caso di operazioni di ingresso e uscita in periferia.

```
FUNCTION_BLOCK FB20
VAR
      raggio_ein: WORD;
      raggio : INT;
END_VAR

BEGIN
raggio_ein  := EB0;
raggio      := WORD_TO_INT(raggio_ein);
(* Conversione in caso di passagio ad un'altra classe di dati.
Il valore proviene dall'ingresso e viene convertito per
eseguire ulteriori calcoli. *)


raggio    := Raggio(superficie:=daticerchio.superficie);
AB0       := WORD_TO_BYTE(INT_TO_WORD(raggio));

(* Raggio viene ricalcolato dalla superficie ed è disponibile
come valore intero. Per l'uscita, il valore viene dapprima
convertito in un'altra classe di dati (INT_TO_WORD) e quindi
nuovamente convertito in un tipo meno importante (WORD_TO_BYTE).
*)
// ...
END_FUNCTION_BLOCK
```

Esempio 18-5 Conversione nel caso di operazioni di ingresso e uscita

Funzioni di arrotondamento e taglio

Le conversioni di tipi di dati comprendono anche le funzioni per arrotondare e tagliare dei numeri. La tabella 18-4 illustra i nomi, i tipi di dati (per il parametro d'ingresso e il valore della funzione) e i compiti di queste funzioni.

Tabella 18-4 Funzioni per arrotondamento e taglio

Nome della funzione	Tipo di dati del parametro d'ingresso	Tipo di dati del valore della funzione	Compito
ROUND	REAL	DINT	Arrotondamento (generazione di un numero DINT)
TRUNC	REAL	DINT	Taglio (generazione di un numero DINT)

Gli esempi seguenti dimostrano i vari modi di funzionamento:

- `ROUND (3.14)     // Qui viene arrotondato`
  `                  // per difetto (Risultato: 3)`
- `ROUND (3.56)     // Qui viene arrotondato`
  `                  // per eccesso (Risultato: 4)`
- `TRUNC (3.14)     // Qui viene tagliato`
  `                  // (Risultato: 3)`
- `TRUNC (3.56)     // Qui viene tagliato`
  `                  // (Risultato: 3)`

18.3 Funzioni standard numeriche

Funzionalità

Ogni funzione standard numerica possiede un parametro d'ingresso. Il risultato è sempre il valore della funzione. Ognuna delle tabelle 18-5, 18-6 e 18-7 specifica un gruppo di funzioni numeriche standard tramite i nomi di funzione e i tipi di dati. Il tipo di dati ANY_NUM indica INT, DINT o REAL.

Funzioni generiche

Si tratta di funzioni per il calcolo del valore assoluto, del quadrato o della radice quadrata di una grandezza.

Tabella 18-5 Funzioni generiche

Nome della funzione	Tipo di dati del parametro d'ingresso	Tipo di dati del valore della funzione	Descrizione
ABS	ANY_NUM[1]	ANY_NUM	Valore assoluto
SQR	ANY_NUM[1]	REAL	Quadrato
SQRT	ANY_NUM[1]	REAL	Radice

1 *Si osservi che i parametri d'ingresso del tipo* `ANY_NUM` *vengono internamente convertiti in variabili REAL.*

Funzioni logaritmiche

Si tratta di funzioni per il calcolo di un valore esponenziale e del logaritmo di una grandezza.

Tabella 18-6 Funzioni logaritmiche

Nome della funzione	Tipo di dati del parametro d'ingresso	Tipo di dati del valore della funzione	Descrizione
EXP	ANY_NUM [1]	REAL	e elevato `IN`
EXPD	ANY_NUM [1]	REAL	10 elevato `IN`
LN	ANY_NUM [1]	REAL	Logaritmo naturale
LOG	ANY_NUM [1]	REAL	Logaritmo decadale

1 *Si osservi che i parametri d'ingresso del tipo* `ANY_NUM` *vengono internamente convertiti in variabili REAL.*

Avvertenza

Si ha inoltre la possibilità di sfruttare le **funzioni IEC** come funzione standard numeriche. In tal caso si deve copiare la funzione desiderata dalla biblioteca STEP 7 STDLIBS\IEC nella propria directory di programma. Per ulteriori informazioni sulle singole funzioni IEC si rimanda a **/235/**.

Funzioni trigonometriche

Le funzioni trigonometriche rappresentate nella tabella 18-7 presuppongono e calcolano grandezze in angoli e archi.

Tabella 18-7 Funzioni trigonometriche

Nome della funzione	Tipo di dati del parametro d'ingresso	Tipo di dati del valore della funzione	Descrizione
ACOS	ANY_NUM[1]	REAL	Arcocoseno
ASIN	ANY_NUM[1]	REAL	Arcoseno
ATAN	ANY_NUM[1]	REAL	Arcotangente
COS	ANY_NUM[1]	REAL	Coseno
SIN	ANY_NUM[1]	REAL	Seno
TAN	ANY_NUM[1]	REAL	Tangente

1 *Si osservi che i parametri d'ingresso del tipo* `ANY_NUM` *vengono internamente convertiti in variabili REAL.*

Esempio

La tabella 18-8 descrive i possibili richiami di funzioni numeriche standard e i rispettivi risultati:

Tabella 18-8 Richiami delle funzioni numeriche standard

Richiamo	Risultato
RISULTATO := ABS (-5);	5
RISULTATO := SQRT (81.0);	9
RISULTATO := SQR (23);	529
RISULTATO := EXP (4.1);	60.340 ...
RISULTATO := EXPD (3);	1_000
RISULTATO := LN (2.718_281);	1
RISULTATO := LOG (245);	2.389_166 ...
PI := 3. 141 592; RISULTATO := SIN (PI / 6);	0.5
RISULTATO := ACOS (0.5);	1.047_197 (=PI / 3)

18.4 Funzioni standard su stringhe di bit

Funzionalità

Ogni funzione standard di stringhe di bit possiede due parametri d'ingresso, identificati da IN e N. Il risultato è sempre il valore della funzione. La tabella 18-9 illustra i nomi delle funzioni e i tipi dei dati dei due parametri d'ingresso e del valore della funzione. Essi significano:

- parametro d'ingresso IN: buffer in cui vengono eseguite le operazioni di scorrimento bit,
- parametro d'ingresso N: numero delle rotazioni per le funzioni del buffer circolare ROL e ROR e numero delle posizioni di bit da far scorrere in SHL e SHR.

Lista di funzioni

La tabella 18-9 illustra le possibili funzioni standard di stringhe di bit:

Tabella 18-9 Funzioni standard di stringhe di bit

Nome della funzione	Tipo di dati del parametro d'ingresso IN	Tipo di dati del parametro d'ingresso N	Tipo di dati del valore della funzione	Compito
ROL	BOOL	INT	BOOL	Il valore presente nel parametro IN viene fatto ruotare verso sinistra di un numero di posizioni di bit indicato nel contenuto del parametro N.
	BYTE	INT	BYTE	
	WORD	INT	WORD	
	DWORD	INT	DWORD	
ROR	BOOL	INT	BOOL	Il valore presente nel parametro IN viene fatto ruotare verso destra di un numero di posizioni di bit indicato nel contenuto del parametro N.
	BYTE	INT	BYTE	
	WORD	INT	WORD	
	DWORD	INT	DWORD	
SHL	BOOL	INT	BOOL	Il valore presente nel parametro IN viene fatto scorrere verso sinistra mentre sul lato destro i bit vengono sostituiti con 0 conformemente al contenuto del parametro N.
	BYTE	INT	BYTE	
	WORD	INT	WORD	
	DWORD	INT	DWORD	
SHR	BOOL	INT	BOOL	Il valore presente nel parametro IN viene fatto scorrere verso destra mentre sul lato sinistro i bit vengono sostituiti con 0 conformemente al contenuto del parametro N.
	BYTE	INT	BYTE	
	WORD	INT	WORD	
	DWORD	INT	DWORD	

Avvertenza

Si ha inoltre la possibilità di sfruttare le **funzioni IEC** per le operazioni di stringhe di bit. In tal caso si deve copiare la funzione desiderata dalla biblioteca STEP 7 STDLIBS\IEC nella propria directory di programma. Per ulteriori informazioni sulle singole funzioni IEC si rimanda a **/235/**.

Esempi

La tabella 18-10 descrive i possibili richiami di funzioni standard di stringhe di bit e i rispettivi risultati.

Tabella 18-10 Richiami delle funzioni standard di stringhe di bit

Richiamo	Risultato
RISULTATO := ROL (IN:=2#1101_0011, N:=5); // IN := 211 decimale	2#0111_1010 (= 122 decimale)
RISULTATO := ROR (IN:=2#1101_0011, N:=2); // IN := 211 decimale	2#1111_0100 (= 244 decimale)
RISULTATO := SHL (IN:=2#1101_0011, N:=3); // IN := 211 decimale	2#1001_1000 (= 152 decimale)
RISULTATO := SHR (IN:=2#1101_0011, N:=2); // IN := 211 decimale	2#0011_0100 (= 52 decimale)

Interfaccia di richiamo

19

Panoramica

Le CPU S7 contengono nel sistema operativo funzioni di sistema e funzioni standard integrate, le quali possono essere di grande aiuto all'utente in fase di programmazione in SCL. Si tratta in particolare di:

- Blocchi organizzativi (OB)
- Funzioni di sistema (SFC)
- Blocchi funzionali di sistema (SFB)

Sommario del capitolo

19.1 Interfaccia di richiamo

Panoramica

Il richiamo dei blocchi può avvenire in modo simbolico o assoluto. A tal fine è necessario un nome simbolico che deve essere stato definito nella tabella dei simboli, oppure il numero per l'identificazione assoluta del blocco.

Per il richiamo, si devono assegnare ai **parametri formali**, i cui nomi e tipi di dati sono stati definiti durante la creazione del blocco parametrizzabile, i **parametri attuali**, con i cui valori il blocco lavora durante l'esecuzione del programma.

Tutte le informazioni necessarie al riguardo sono descritte nel manuale di riferimento **/235/**. Il manuale fornisce una panoramica delle funzioni disponibili in S7 e – come informazioni a titolo di consultazione – delle descrizioni dettagliate delle interfacce per l'utilizzo nel proprio programma utente.

Esempio di SFC 31

Le seguenti righe di comando consentono il richiamo della funzione di sistema SFC 31 (Interroga allarme dall'orologio):

```
FUNCTION_BLOCK FY20

VAR
      Risultato: INT;
END_VAR
BEGIN
      // ...
      RISULTATO:= SFC31 (OB_NR:= 10,STATUS:= MW100 );
      // ...
      // ...
END_FUNCTION_BLOCK
```

Esempio 19-1 Interrogazione dell'allarme dall'orologio

Risultati

Il valore della funzione è di tipo integer. Se il suo valore è > = 0, il blocco è stato eseguito senza errori, mentre se il valore è < 0, si è verificato un errore. Dopo il richiamo si può analizzare il parametro d'uscita ENO definito implicitamente.

Richiamo condizionato

Per un richiamo condizionato, si deve impostare a 0 il **parametro d'ingresso EN** predefinito (p. es. tramite l'ingresso E0.3); in questo caso il blocco non viene richiamato. Se EN viene impostato a 1, la funzione viene richiamata. In tal caso anche il **parametro d'uscita ENO** viene impostato a "1" (altrimenti a "0"), se durante l'elaborazione del blocco non si verifica alcun errore.

Avvertenza

Nei blocchi funzionali o nei blocchi funzionali di sistema, le informazioni che si possono trasferire nel caso di una funzione con il valore della funzione devono essere memorizzate in parametri d'uscita. Questi parametri d'uscita vengono quindi analizzati tramite i blocchi dati di istanza. Per ulteriori informazioni si rimanda al capitolo 16.

19.2 Interfaccia di trasferimento agli OB

Blocchi organizzativi

I blocchi organizzativi costituiscono l'interfaccia fra il sistema operativo della CPU e il programma utente. Con l'ausilio di OB si possono eseguire determinate parti di un programma:

- all'avviamento della CPU
- con esecuzione a frequenza ciclica o temporale
- a determinate ore o in determinati giorni
- allo scadere di una durata di tempo predefinita
- se si verificano degli errori
- se si verificano interrupt di processo o allarmi di comunicazione

I blocchi organizzativi vengono elaborati in base alla priorità loro assegnata.

OB disponibili

Non tutte le CPU sono in grado di elaborare tutti gli OB disponibili in S7. Per una descrizione degli OB disponibili si rimanda ai dati tecnici della CPU usata.

Ulteriori informazioni

Per ulteriori informazioni si rimanda alla guida online e ai seguenti manuali:

- **/70/** Manuale: *Sistema di automazione S7-300,* Installazione, configurazione e dati della CPU. Questo manuale contiene i dati tecnici che descrivono il volume di prestazioni delle varie CPU S7-300. Fra queste ultime vi sono anche i possibili eventi di avvio per ogni OB.
- **/100/** Manuale: *Sistemi di automazione S7-400, M7-400,* Configurazione. Questo manuale contiene i dati tecnici che descrivono il volume di prestazioni delle varie CPU S7-400. Fra queste ultime vi sono anche i possibili eventi di avvio per ogni OB.

Appendici

Descrizione formale del linguaggio

Descrizione del linguaggio SCL

I diagrammi sintattici costituiscono la base per la descrizione del linguaggio nei singoli capitoli. Tali diagrammi forniscono una panoramica della struttura sintattica (ossia grammaticale) di SCL. Gli appendici B e C contengono un elenco completo di tutti i diagrammi con gli elementi linguistici.

Sommario dell'appendice

A.1 Panoramica

Che cos'è un diagramma sintattico?

Il diagramma sintattico è una rappresentazione grafica della struttura del linguaggio. La struttura viene descritta da una sequenza di regole. Una regola può anche basarsi su altre regole già definite.

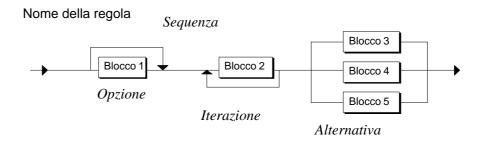



Figura A-1 Esempio del diagramma sintattico

Il diagramma sintattico viene letto da sinistra verso destra. Si devono osservare le seguenti strutture delle regole:

- Sequenza: sequenza di blocchi
- Opzione: diramazione saltabile
- Iterazione: ripetizione di diramazioni
- Alternativa: diramazione

Quali tipi di blocchi esistono?

Un blocco è un elemento base oppure un elemento che a sua volta si compone di blocchi. La figura seguente illustra i tipi di simboli corrispondenti ai blocchi:

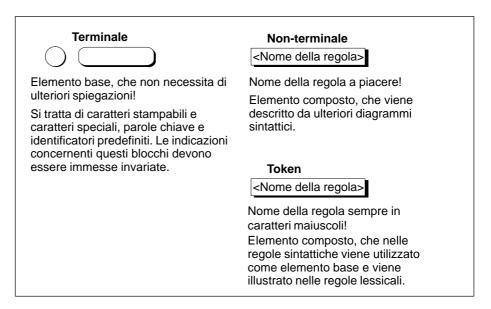



Figura A-2 Tipi di simboli dei blocchi

Regole

Le regole che possono essere utilizzate per creare il programma utente SCL sono suddivise in regole **lessicali** e regole **sintattiche**.

Regole lessicali

Le regole lessicali descrivono la struttura degli elementi (token) che vengono elaborati durante l'analisi lessicale del compilatore. Perciò, la modalità di scrittura non è in formato libero, cioè le regole devono essere rigorosamente rispettate. Ciò significa in modo particolare:

- non è consentito inserire caratteri di formattazione
- non è consentito inserire commenti al blocco e alla riga
- non è consentito inserire attributi per gli identificatori

IDENTIFICATORE

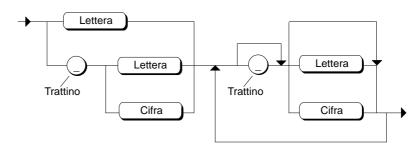



Figura A-3 Esempio di regola lessicale

L'esempio illustra la regola lessicale IDENTIFICATORE. Essa descrive la struttura di un identificatore (nome), p. es.:

```
CAMPO_MIS_12
VALORERIF_B_1
```

Regole sintattiche

Sulla base delle regole lessicali, le regole sintattiche descrivono la struttura di SCL. Nell'ambito di tali regole l'utente può creare il proprio programma SCL in formato libero:

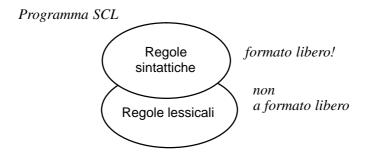



Figura A-4 Livelli delle regole e libertà di formato

Formalità

Ogni regola possiede un nome che viene preposto. Se la regola viene utilizzata in una regola sovraordinata, questo nome appare in un rettangolo.

Se il nome della regola è scritto in caratteri maiuscoli, si tratta di un token che viene descritto nelle regole lessicali.

Semantica

Nelle regole può essere descritta solo la struttura formale del linguaggio. Esse non sempre descrivono il significato, cioè la semantica. Perciò, nei punti più importanti, accanto alle regole vengono scritte sempre informazioni supplementari. Seguono alcuni esempi di queste informazioni.

- In caso di elementi dello stesso tipo, ma con significato diverso viene indicato un nome supplementare: p. es. nella regola dell'indicazione della data nella SEQUENZA DI CIFRE DECIMALI anno, mese o giorno. Il nome indica il tipo d'impiego.
- Le restrizioni importanti vengono annotate accanto alle regole: p. es. per il simbolo viene annotato che esso deve essere definito nella tabella dei simboli.

A.2 Panoramica dei terminali

Definizione

Un terminale è un elemento base che non viene spiegato con un'ulteriore regola, ma in modo verbale. Nei diagrammi sintattici esso viene indicato con il simbolo seguente:

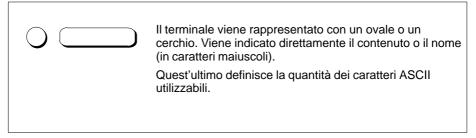



Figura A-5 Simboli per terminali

Panoramica

Nei capitoli A.3 fino a A.4 vengono illustrate le modalità d'uso dei singoli caratteri. Si tratta di:

- Lettere, cifre, caratteri stampabili e caratteri speciali
- Caratteri di formattazione e caratteri di separazione nelle regole lessicali
- Prefissi per literal
- Caratteri di formattazione e caratteri di separazione nelle regole sintattiche
- Operatori

I capitoli A.5 e A.6 trattano parole chiave e identificatori predefiniti da stringhe di caratteri. Le tabelle sono elencate in ordine alfabetico. In caso di differenze fra il mnemonico SIMATIC e il mnemonico internazionale viene indicato anche il corrispondente mnemonico internazionale (IEC).

- Parole chiave e identificatori predefiniti
- Identificazioni di operandi e parole chiave di blocchi

A.3 Terminali delle regole lessicali

Panoramica

Nelle tabelle seguenti, i terminali sono specificati mediante indicazione della quantità degli elementi del set di caratteri ASCII.

Lettere e cifre

Sono i caratteri principalmente utilizzati. L'IDENTIFICATORE si compone p. es. di lettere, cifre e del trattino.

Tabella A-1 Lettere e cifre

Carattere	Sottogruppo	Elementi del set di caratteri
Lettera	Carattere maiuscolo	A.. Z
	Carattere minuscolo	a.. z
Cifra	Cifra decimale	0.. 9
Cifra ottale	Cifra ottale	0.. 7
Cifra esadecimale	Cifra esadecimale	0.. 9, A.. F, a.. f
Bit	Cifra binaria	0, 1

Caratteri stampabili e caratteri speciali

Il set di caratteri ASCII ampliato e completo può essere utilizzato in stringhe, commenti e simboli.

Tabella A-2 Caratteri stampabili e caratteri speciali

Carattere	Sottogruppo	Elementi del set di caratteri
Carattere stampabile	Dipende dal codice di carattere utilizzato. Per esempio nel codice ASCII a partire dall'equivalente decimale 31, senza DEL e senza i seguenti caratteri sostitutivi:	Tutti i caratteri stampabili
Carattere sostitutivo	Carattere del dollaro	$
	Apostrofo	'
Carattere di controllo	$P o $p	Rimpaginazione (formfeed, page)
	$L o $l	A capo automatico (linefeed)
	$R o $r	Ritorno del carrello (carriage return)
	$T o $t	Tabulatore
Rappresentazione sostitutiva in codice esadecimale	$hh	Qualsiasi carattere, rappresentabile in codice esadecimale (hh)

A.4 Caratteri di formattazione, separatori e operatori

Nelle regole lessicali

La tabella A-3 descrive l’impiego di singoli caratteri del set di caratteri ASCII come caratteri di formattazione, separatori e operatori nell’ambito delle regole lessicali (vedere appendice B).

Tabella A-3 Caratteri di formattazione e separatori nelle regole lessicali

Carattere	Descrizione
:	Separatori fra ore, minuti e secondi Attributi
.	Separatori per indirizzamento assoluto e rappresentazione di numeri in virgola mobile e intervalli di tempo
’ ’	Caratteri e stringhe di caratteri
” ”	Caratteri introduttivi per simboli in base alle regole della tabella dei simboli
_ Trattino	Separatore per valori numerici in literal, può essere presente in IDENTIFICATORI
$	Simbolo del dollaro per indicare caratteri di controllo o caratteri sostitutivi
$> $<	Simbolo di interruzione stringa nel caso in cui una stringa non rientri in una riga o se si intendono inserire dei commenti.

Per literal

La tabella A-4 descrive l’uso di singoli caratteri e stringhe di caratteri per literal nell’ambito delle regole lessicali (vedere appendice B). La tabella vale sia per il mnemonico SIMATIC sia per quello IEC.

Tabella A-4 Mnemonico per literal, in ordine alfabetico

Prefisso	Contrassegno per	Regola lessicale
2#	LITERAL INTEGER	Sequenza di cifre binarie
8#	LITERAL INTEGER	Sequenza di cifre ottali
16#	LITERAL INTEGER	Sequenza di cifre esadecimali
D#	Indicazione della data	DATA
DATE#	Indicazione della data	DATA
DATE_AND_TIME#	Indicazione della data e dell’ora	DATA E ORA
DT#	Indicazione della data e dell’ora	DATA E ORA
E	Separatore per LITERAL NUMERO REALE	Esponente
e	Separatore per LITERAL NUMERO REALE	Esponente
D	Separatore per intervallo di tempo (Day)	Giorni (Regola: rappresentazione a livelli)
H	Separatore per intervallo di tempo (Hour)	Ore: (Regola: rappresentazione a livelli)
M	Separatore per intervallo di tempo (Minutes)	Minuti : (Regola: rappresentazione a livelli)
MS	Separatore per intervallo di tempo (Milliseconds)	Millisecondi: (Regola: rappresentazione a livelli)
S	Separatore per intervallo di tempo (Seconds)	Secondi: (Regola: rappresentazione a livelli)
T#	Indicazione del tempo	DURATA

Tabella A-4 Mnemonico per literal, in ordine alfabetico

Prefisso	Contrassegno per	Regola lessicale
TIME#	Indicazione del tempo	DURATA
TIME_OF_DAY#	Indicazione del tempo	ORA DEL GIORNO
TOD#	Indicazione del tempo	ORA DEL GIORNO

Nelle regole sintattiche

La tabella A-5 descrive l'impiego di singoli caratteri del set di caratteri ASCII come caratteri di formattazione e separatori nell'ambito delle regole sintattiche, nei commenti e negli attributi (vedere appendice B.2 e B.3).

Tabella A-5 Caratteri di formattazione e separatori nelle regole sintattiche

Carattere	Descrizione	Regola sintattica, commento o attributo
:	Separatore per indicazione del tipo, con istruzione dopo etichetta di salto	Dichiarazione di variabile, dichiarazione di istanza, funzione, parte istruzioni, istruzione CASE
;	Conclusione di una convenzione o istruzione	Dichiarazione di costante e di variabile, parte istruzioni, parte assegnazioni, blocco costanti, blocco etichette di salto
,	Separatore per liste e blocchi di etichette di salto	Dichiarazione di variabile, specificazione tipo di dati array, lista inizializzazione di campo, parametro di FB, parametro di FC, lista di valori, dichiarazione di istanza
..	Indicazione di campo	Specificazione tipo di dati array, lista di valori
.	Separatore per nome di FB e DB, indirizzamento assoluto	Richiamo di FB, variabile strutturata
( )	Richiamo di funzione e di blocco funzionale, parentesi in espressioni, lista di inizializzazione per array	Richiamo di funzione, richiamo di FB, espressione, lista inizializzazione di campo, moltiplicazione semplice, espressione di potenza
[ ]	Dichiarazione di array, variabile strutturata parte array, indicizzazione nelle variabili globali e nelle stringhe	Specificazione tipo di dati array, specificazione tipo di dati STRING
(* *)	commento a più righe	vedere appendice B
//	commento a una riga	vedere appendice B
{ }	Blocco di attributo	Specificazione di attributi

Operatori

Nella tabella A-6 sono descritti tutti gli operatori SCL, la parole chiave, p. es. AND e gli operatori usuali come caratteri singoli. La tabella è valida per il mnemonico SIMATIC e IEC.

Tabella A-6 Operatori SCL

Operatore	Descrizione	Esempio di regola sintattica
:=	Operatore di assegnazione, assegnazione iniziale, inizializzazione del tipo di dati	Assegnazione di valori, parte assegnazioni DB, blocco costanti, assegnazione d'uscita/di transito, assegnazione d'ingresso, assegnazione di transito
+, -	Operatori aritmetici: operatori unari, segno	Espressione, espressione semplice, espressione di potenza

Tabella A-6 Operatori SCL

Operatore	Descrizione	Esempio di regola sintattica
+, -, *, / MOD; DIV	Operatori aritmetici di base	Operatore aritmetico di base, moltiplicazione semplice
**	Operatori aritmetici: operatore di potenza	Espressione
NOT	Operatori logici: negazione	Espressione
AND, &, OR; XOR,	Operatori logici di base	Operatore logico di base
<,>,<=,>=,=,<>	Operatori di confronto	Operatore di confronto

A.5 Parole chiave e identificatori predefiniti

Parole chiavi e identificatori predefiniti

La tabella A-7 contiene parole chiave e identificatori predefiniti di SCL, elencati in ordine alfabetico. Vengono inoltre riportate descrizioni e regole sintattiche dell'appendice C, in cui esse vengono utilizzate come terminali. In generale, le parole chiave non dipendono dal mnemonico.

Tabella A-7 Parole chiave SCL e identificatori predefiniti, in ordine alfabetico

Parole chiave	Descrizione	Regola sintattica
AND	Operatore logico	Operatore logico di base
ANY	Identificazione per il tipo di dati ANY	Specificazione tipo di dati parametro
ARRAY	Introduzione della specificazione di un array, segue la lista indice fra ”[” e ”]”	Specificazione tipo di dati array
BEGIN	Introduzione parte istruzioni nel blocco di codice o nella parte inizializzazione del blocco dati	Blocco organizzativo, funzione, blocco funzionale, blocco dati
BLOCK_DB	Identificazione per il tipo di dati BLOCK_DB	Specificazione del tipo di dati parametro
BLOCK_FB	Identificazione per il tipo di dati BLOCK_FB	Specificazione del tipo di dati parametro
BLOCK_FC	Identificazione per il tipo di dati BLOCK_FC	Specificazione del tipo di dati parametro
BLOCK_SDB	Identificazione per il tipo di dati BLOCK_SDB	Specificazione del tipo di dati parametro
BOOL	Tipo di dati semplice per dati binari	Tipo di dati bit
BY	Introduzione dell'ampiezza di passo	Istruzione FOR
BYTE	Tipo di dati semplice	Tipo di dati bit
CASE	Introduzione istruzione di controllo per selezione	Istruzione CASE
CHAR	Tipo di dati semplice	Tipo del carattere
CONST	Introduzione per la definizione di costanti	Blocco costanti
CONTINUE	Istruzione di controllo per loop FOR, WHILE e REPEAT	Istruzioni CONTINUE
COUNTER	Tipo di dati per contatori, utilizzabile solo nel blocco parametri	Specificazione del tipo di dati parametro
DATA_BLOCK	Introduzione di blocco dati	Blocco dati
DATE	Tipo di dati semplice per data	Tipo di tempo
DATE_AND_TIME	Tipo di dati composto per data e ora	DATE_AND_TIME
DINT	Tipo di dati semplice per numero intero (Integer) doppia precisione	Tipo di dati numerici
DIV	Operatore per divisione	Operatore base aritmetico, moltiplicazionesemplice
DO	Introduzione parte istruzioni nell'istruzione FOR	Istruzione FOR, istruzione WHILE
DT	Tipo di dati semplice per data e ora	DATE_AND_TIME
DWORD	Tipo di dati semplice doppia parola	Tipo di dati bit

Tabella A-7 Parole chiave SCL e identificatori predefiniti, in ordine alfabetico

Parole chiave	Descrizione	Regola sintattica
ELSE	Introduzione nel caso se non viene soddisfatta alcuna condizione	Istruzione IF
ELSIF	Introduzione della condizione alternativa	Istruzione IF
EN	Flag per abilitazione blocco	
ENO	Flag di errore del blocco	
END_CASE	Fine dell'istruzione CASE	Istruzione CASE
END_CONST	Fine della definizione di costanti	Blocco costanti
END_DATA_BLOCK	Fine del blocco dati	Blocco dati
END_FOR	Fine dell'istruzione FOR	Istruzione FOR
END_FUNCTION	Fine della funzione	Funzione
END_FUNCTION_BLOCK	Fine del blocco funzionale	Blocco funzionale
END_IF	Fine dell'istruzione IF	Istruzione IF
END_LABEL	Fine della dichiarazione di un blocco etichette di salto	Blocco etichette di salto
END_TYPE	Fine dell'UDT	Tipo di dati definito dall'utente
END_ORGANIZATION_BLOCK	Fine del blocco organizzativo	Blocco organizzativo
END_REPEAT	Fine dell'istruzione REPEAT	Istruzione REPEAT
END_STRUCT	Fine della specificazione di una struttura	Specificazione tipo di dati struttura
END_VAR	Fine del blocco di dichiarazione	Blocco variabili temporanee, blocco variabili statiche, blocco parametri
END_WHILE	Fine dell'istruzione WHILE	Istruzione WHILE
EXIT	Uscita diretta dall'elaborazione di loop	EXIT
FALSE	Costante booleana predefinita: condizione logica non soddisfatta, valore uguale a 0	
FOR	Introduzione dell'istruzione di controllo per l'elaborazione di loop	Istruzione FOR
FUNCTION	Introduzione della funzione	Funzione
FUNCTION_BLOCK	Introduzione del blocco funzionale	Blocco funzionale
GOTO	Istruzione per l'esecuzione di un salto ad un'etichetta di salto	Salto di programma
IF	Introduzione istruzione di controllo per selezione	Istruzione IF
INT	Tipo di dati semplice per numero intero (Integer), semplice precisione	Tipo di dati numerici
LABEL	Introduzione per dichiarazione di un blocco etichette di salto	Blocco etichette di salto
MOD	Operatore aritmetico per resto della divisione	Operatore base aritmetico, moltiplicazione semplice
NIL	Puntatore zero	
NOT	Operatore logico, fa parte degli operatori unari	Espressione, operando
OF	Introduzione della specificazione tipo di dati	Specificazione tipo di dati array, istruzione CASE

Tabella A-7 Parole chiave SCL e identificatori predefiniti, in ordine alfabetico

Parole chiave	Descrizione	Regola sintattica
OK	Flag indicante se le istruzioni di un blocco sono state elaborate senza errori	
OR	Operatore logico	Operatore base logico
ORGANIZATION_ BLOCK	Introduzione del blocco organizzativo	Blocco organizzativo
POINTER	Tipo di dati puntatore, consentito solo nella dichiarazione parametri del blocco parametri, non viene elaborato in SCL	Vedere capitolo 10
REAL	Tipo di dati semplice	Tipo di dati numerici
REPEAT	Introduzione dell'istruzione di controllo per l'elaborazione di loop	Istruzione REPEAT
RETURN	Istruzione di controllo per il ritorno dal sottoprogramma	Istruzione RETURN
S5TIME	Tipo di dati semplice per indicazioni di tempo, formato speciale S5	Tipo di tempo
STRING	Tipo di dati per stringa di caratteri	Specificazione tipo di dati STRING
STRUCT	Introduzione della specificazione di una struttura, segue la lista dei componenti	Specificazione tipo di dati struttura
THEN	Introduzione delle azioni seguenti, se è soddisfatta la condizione	Istruzione IF
TIME	Tipo di dati semplice per indicazioni di tempo	Tipo di tempo
TIMER	Tipo di dati per temporizzatore, utilizzabile solo nel blocco parametri	Specificazione tipo di dati parametro
TIME_OF_DAY	Tipo di dati semplice per ora del giorno	Tipo di tempo
TO	Introduzione del valore finale	Istruzione FOR
TOD	Tipo di dati semplice per ora del giorno	Tipo di tempo
TRUE	Costante booleana predefinita: condizione logica soddisfatta, valore diverso da 0	
TYPE	Introduzione dell'UDT	Tipo di dati definito dall'utente
VAR	Introduzione blocco di dichiarazione	Blocco variabili statiche
VAR_TEMP	Introduzione blocco di dichiarazione	Blocco variabili temporanee
UNTIL	Introduzione della condizione d'interruzione per l'istruzione REPEAT	Istruzione REPEAT
VAR_INPUT	Introduzione blocco di dichiarazione	Blocco parametri
VAR_IN_OUT	Introduzione blocco di dichiarazione	Blocco parametri
VAR_OUTPUT	Introduzione blocco di dichiarazione	Blocco parametri
WHILE	Introduzione dell'istruzione di controllo per l'elaborazione di loop	Istruzione WHILE
WORD	Tipo di dati semplice parola	Tipo di dati bit
VOID	Nessun valore di ritorno in un richiamo di funzione	Funzione
XOR	Operatore logico	Operatore logico

A.6 Identificazioni di operando e parole chiavi del blocco

Dati globali del sistema

La tabella A-8 contiene il mnemonico SIMATIC delle identificazioni di operando di SCL con descrizione in ordine alfabetico:

- Indicazione dell'identificazione di operando:
  Prefisso di memoria (A, E, M, PA, PE) o blocco dati (D)
- Indicazione della dimensione dell'elemento dati:
  Prefisso di dimensione (opzionale o B, D, W, X)

Il mnemonico rappresenta una combinazione fra l'identificazione di operando (prefisso di memoria o D per blocco dati) e prefisso di dimensione. Entrambi sono regole lessicali. La tabella è ordinata secondo il mnemonico SIMATIC e viene indicata anche il corrispondente mnemonico IEC.

Tabella A-8 Identificazioni di operando dei dati globali del sistema

Mnemonico SIMATIC	Mnemonico IEC	Prefisso di memoria o blocco dati	Prefisso di dimensione
A	Q	Uscita (tramite immagine di processo)	Bit
AB	QB	Uscita (tramite immagine di processo)	Byte
AD	QD	Uscita (tramite immagine di processo)	Doppia parola
AW	QW	Uscita (tramite immagine di processo)	Parola
AX	QX	Uscita (tramite immagine di processo)	Bit
D	D	Blocco dati	Bit
DB	DB	Blocco dati	Byte
DD	DD	Blocco dati	Doppia parola
DW	DW	Blocco dati	Parola
DX	DX	Blocco dati	Bit
E	I	Ingresso (tramite immagine di processo)	Bit
EB	IB	Ingresso (tramite immagine di processo)	Byte
ED	ID	Ingresso (tramite immagine di processo)	Doppia parola
EW	IW	Ingresso (tramite immagine di processo)	Parola
EX	IX	Ingresso (tramite immagine di processo)	Bit
M	M	Merker	Bit
MB	MB	Merker	Byte
MD	MD	Merker	Doppia parola
MW	MW	Merker	Parola
MX	MX	Merker	Bit
PAB	PQB	Uscita (periferia diretta)	Byte
PAD	PQD	Uscita (periferia diretta)	Doppia parola
PAW	PQW	Uscita (periferia diretta)	Parola
PEB	PIB	Ingresso (periferia diretta)	Byte

Tabella A-8 Identificazioni di operando dei dati globali del sistema

Mnemonico SIMATIC	Mnemonico IEC	Prefisso di memoria o blocco dati	Prefisso di dimensione
PED	PID	Ingresso (periferia diretta)	Doppia parola
PEW	PIW	Ingresso (periferia diretta)	Parola

Parole chiave del blocco

Vengono utilizzate per l'indirizzamento assoluto di blocchi. La tabella è ordinata in base al mnemonico SIMATIC e viene indicata anche il corrispondente mnemonico IEC.

Tabella A-9 Parole chiavi del blocco nonché contatori e temporizzatori

Mnemonico SIMATIC	Mnemonico IEC	Prefisso di memoria o blocco dati
DB	DB	Blocco dati (Data-Block)
FB	FB	Blocco funzionale (Function Block)
FC	FC	Funzione (Function)
OB	OB	Blocco organizzativo (Organization Block)
SDB	SDB	Blocco dati di sistema (System Data Block)
SFC	SFC	Funzione di sistema (System Function)
SFB	SFB	Blocco funzionale di sistema (System Function Block)
T	T	Temporizzatore (Timer)
UDT	UDT	Tipo di dati globale o definito dall'utente (Userdefined Data Type)
Z	C	Contatore (Counter)

A.7 Panoramica dei non-terminali

Definizione

Un non-terminale è un elemento composto che viene descritto da un'ulteriore regola. Il non-terminale viene rappresentato da quadrato. Il nome nel quadrato corrisponde al nome della regola successiva.

Figura A-6 Non-terminale

L'elemento è presente nelle regole lessicali e nelle regole sintattiche.

A.8 Panoramica dei token

Definizione

Un token è un elemento composto impiegato come elemento base nelle regole sintattiche e viene dichiarato nelle regole lessicali. Il token viene rappresentato da un rettangolo. Il NOME, in caratteri maiuscoli, corrisponde al nome della regola lessicale seguente (senza rettangolo).

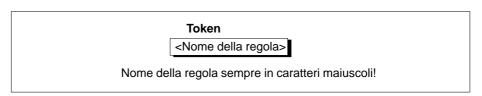



Figura A-7 Token

Panoramica

I token definiti rappresentano identificatori che sono stati determinati come risultato delle regole lessicali. Questi token descrivono:

- Identificatori
- Assegnazioni di nomi in SCL
- Costanti predefinite e flag

A.9 Identificatori

Identificatori in SCL

Gli identificatori consentono di far riferimento agli oggetti linguistici di SCL. La tabella A-10 contiene informazioni sulle classi di identificatori:

Tabella A-10 Elenco complessivo dei tipi di identificatori in SCL

Tipo di identificatore	Note, esempi
Parole chiave	p. es. Istruzioni di controllo `BEGIN, DO,WHILE`
Nomi predefiniti	Nomi di • tipo di dati standard (p. es. BOOL, BYTE, INT) • funzioni standard predefinite p. es. ABS • costanti standard `TRUE e FALSE`
Identificazioni di operandi negli identificatori assoluti	per dati globali del sistema e blocchi di dati: p. es. `E1.2, MW10, FC20, T5, DB30, DB10.D4.5`
Nomi selezionabili liberamente secondo la regola IDENTIFICATORE	Nomi di • variabili dichiarate • componenti di strutture • parametri • costanti dichiarate • etichette di salto
Simboli della tabella dei simboli	soddisfano la regola lessicale IDENTIFICATORE o la regola lessicale simbolo, cioè racchiuse fra virgolette, p. es. ”xyz”

Maiuscole/ minuscole

Per quanto riguarda le parole chiave, le minuscole e le maiuscole non sono rilevanti. Dalla versione 4.0 di S7 SCL non viene più fatta distinzione tra maiuscole e minuscole, né nei nomi predefiniti né in quelli a libera scelta, p. es. le variabili, e neppure nei simboli della tabella dei simboli. La tabella A-11 ne fornisce una visione di insieme.

Tabella A-11 Rilevanza delle maiuscole/minuscole nei vari tipi di identificatori

Tipo di identificatore	case-sensitiv?
Parole chiave	no
Nomi predefiniti nei tipi di dati standard	no
Nomi nelle funzioni standard predefinite	si
Nomi predefiniti nelle costanti standard	no
Identificazioni di operando negli identificatori assoluti	no
Nomi liberi	si
Simboli della tabella dei simboli	si

I nomi delle funzioni standard, p. es. BYTE_TO_WORD e ABS, possono così essere scritti anche a lettere minuscole. Allo stesso modo i parametri per le funzioni di temporizzazione e di conteggio, p. es. SV, se o ZV.

A.10 Assegnazione di nomi in SCL

Assegnazione di nomi selezionabili

In genere, per l'assegnazione di nomi si hanno due possibilità:

- Si possono assegnare dei nomi nell'ambito di SCL. Questi nomi devono corrispondere alla regola IDENTIFICATORE in base alla figura A-8. La regola IDENTIFICATORE può essere usata per ogni nome in SCL.
- Inoltre, si possono inserire nomi tramite STEP 7 con l'ausilio della tabella dei simboli. Anche per questi nomi la regola è IDENTIFICATORE o Simbolo come ulteriore possibilità. Grazie alle virgolette, il simbolo può venire formato da tutti i caratteri stampabili (p. es. spazio).

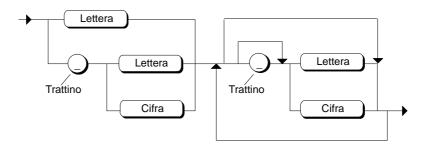
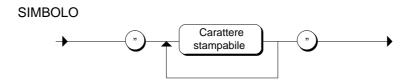
IDENTIFICATORE

SIMBOLO

Figura A-8 Regole lessicali IDENTIFICATORE e Simbolo

Regole per l'assegnazione dei nomi

Si prega di tener presente quanto segue:

- Per l'assegnazione dei nomi si consiglia di scegliere nomi univoci ed espressivi che facilitano la comprensione del programma.
- Si deve verificare dapprima se il nome è già occupato dal sistema, p. es. da identificatori dei tipi di dati o delle funzioni standard.
- Campo di validità: per i nomi che hanno validità globale, il campo di validità si estende per l'intero programma. I nomi con validità locale sono validi solo all'interno di un blocco. In tal modo si ha la possibilità di utilizzare lo stesso nome in diversi blocchi. La tabella A-12 informa l'utente sulle varie alternative.

Restrizioni per l'assegnazione di nomi

Durante l'assegnazione di nomi si devono osservare le seguenti restrizioni:

I nomi devono essere univoci nel loro campo di validità, cioè i nomi che sono già stati assegnati all'interno di un blocco non possono essere più assegnati nello stesso blocco. Inoltre, i seguenti nomi occupati dal sistema non possono essere utilizzati:

- Nomi di parole chiavi: p. es. `CONST, END_CONST, BEGIN`
- Nomi di operatori: p. es. `AND, XOR`
- Nomi di identificatori predefiniti: p. es. nomi per tipi di dati come `BOOL, STRING, INT`
- Nomi di costanti predefinite `TRUE` e `FALSE`
- Nomi di funzioni standard: p. es. `ABS, ACOS, ASIN, COS, LN`
- Nomi di identificazioni assoloute o di operandi per dati globali del sistema: p. es. `EB, EW, ED, AB, AW, AD MB, MD`

Impiego di IDENTIFICATORE

La tabella A-12 indica in quali casi l'utente può indicare nomi corrispondenti alla regola IDENTIFICATORE.

Tabella A-12 Uso di IDENTIFICATORE

IDENTIFICA-TORE	Descrizione	Regola
Nome del blocco	Nome simbolico per blocchi	IDENTIFICAZIONE BLOCCO, richiamo di funzione
Nomi per temporizzatori e contatori	Nome simbolico per temporizzatori e contatori	IDENTIFICAZIONE TEMPORIZZATORE, IDENTIFICAZIONE CONTATORE
Nome di attributo	Nome per un attributo	Assegnazione di attributo
Nome di costante	Dichiarazione costante simbolica, impiego	Blocco costanti, costante
Etichetta di salto	Dichiarazione di etichetta di salto, impiego etichetta di salto	Parte istruzioni del blocco etichette di salto, istruzione GOTO
Nome di variabile	Dichiarazione di variabile temporanea o statica	Dichiarazione di variabile, Variabile semplice, Variabile strutturata
Nome di istanza locale	Dichiarazione di istanza locale	Dichiarazione di istanza, nome di richiamo di FB

IDENTIFICAZIONE BLOCCO

La regola IDENTIFICAZIONE BLOCCO è un caso in cui IDENTIFICATORE e simbolo possono essere impiegati in modo alternativo:

IDENTIFICAZIONE BLOCCO

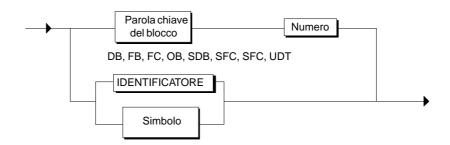



Figura A-9 Regola lessicale IDENTIFICAZIONE BLOCCO

Le regole IDENTIFICAZIONE TEMPORIZZATORE e IDENTIFICAZIONE CONTATORE sono analoghe a IDENTIFICAZIONE BLOCCO. Per tali regole valgono gli stessi criteri.

A.11 Costanti predefinite e flag

Costanti predefinite e flag

Entrambe le tabelle valgono per il mnemonico SIMATIC e il mnemonico IEC.

Tabella A-13 Costanti predefinite

Mnemonico	Descrizione
FALSE	Costante booleana predefinita (costante standard) con il valore 0. Il suo significato logico è che la condizione non è soddisfatta.
TRUE	Costante booleana predefinita (costante standard) con il valore 1. Il suo significato logico è che la condizione è soddisfatta.

Tabella A-14 Flag

Mnemonico	Descrizione
EN	Flag per l’abilitazione del blocco
ENO	Flag di errore del blocco
OK	Viene impostato su FALSE quando un’istruzione è stata eseguita in modo errato.

Regole lessicali

B

Sommario dell'appendice

Regole lessicali

Le regole lessicali descrivono la struttura degli elementi (token) che vengono elaborati durante l'analisi lessicale del compilatore. Perciò, il modo di scrittura non ha un formato libero, ma si devono rispettare rigorosamente determinate regole. Ciò significa in particolare:

- Non è consentito inserire caratteri di formattazione
- Non si possono inserire commenti a una riga e a più righe
- Non si possono inserire attributi per gli identificatori

Suddivisione

Le regole lessicali sono suddivise nei gruppi seguenti:

- Identificatori
- Literal
- Indirizzamento assoluto

B.1 Identificatori

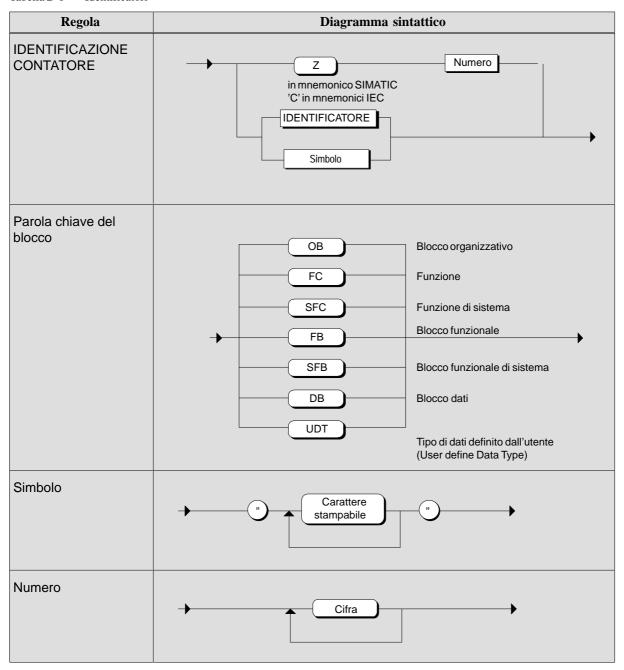



Tabella B-1 Identificatori

Tabella B-1 Identificatori

B.1.1 Literal

Tabella B-2 Literal

Tabella B-2 Literal

Tabella B-2 Literal

Tabella B-2 Literal

Regola	Diagramma sintattico
Indicazione della data	SEQUENZA DI CIFRE DECIMALI (Anno) – SEQUENZA DI CIFRE DECIMALI (Mese) – SEQUENZA DI CIFRE DECIMALI (Giorno)
Indicazione dell'ora del giorno	SEQUENZA DI CIFRE DECIMALI (Indicazione delle ore) : SEQUENZA DI CIFRE DECIMALI (Inicazione dei minuti) : SEQUENZA DI CIFRE DECIMALI (Indicazione dei millisecondi) . SEQUENZA DI CIFRE DECIMALI (Indicazione dei millisecondi)

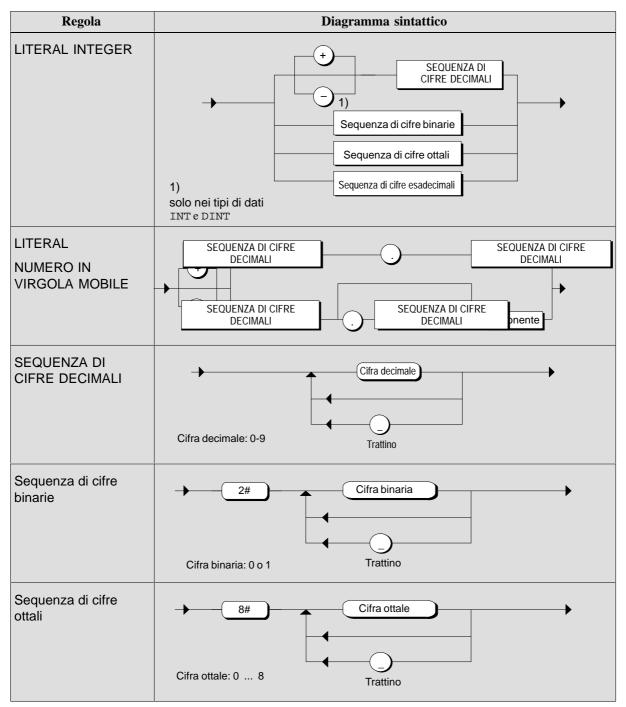
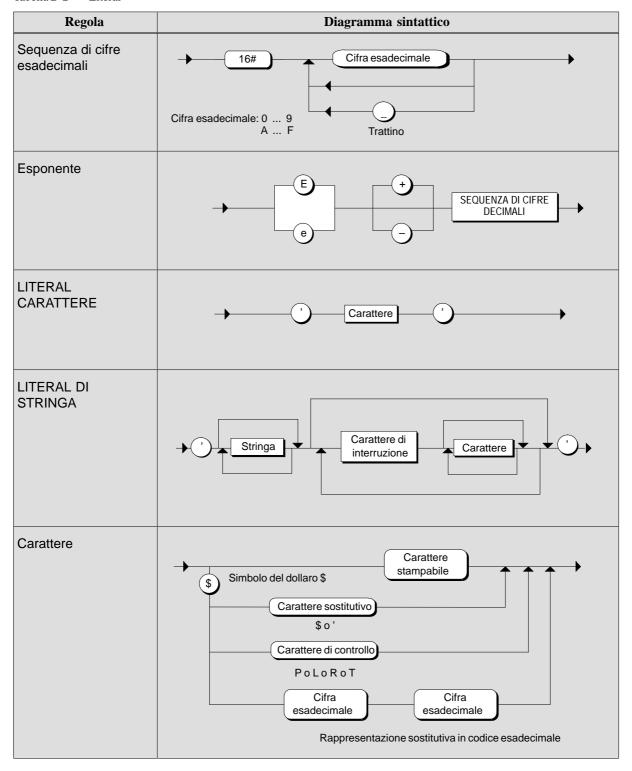
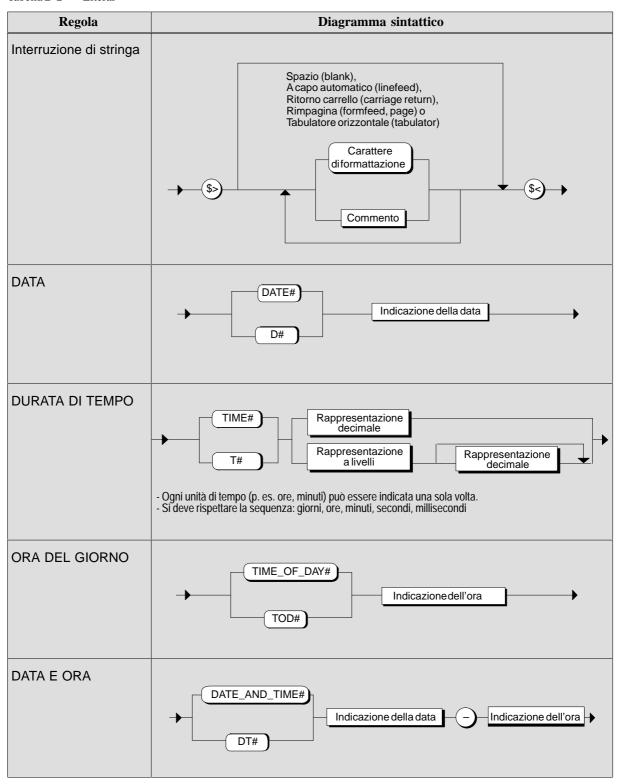
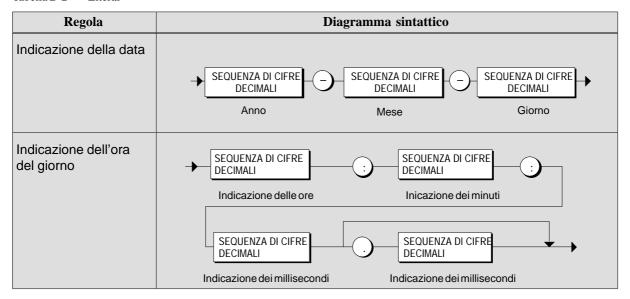



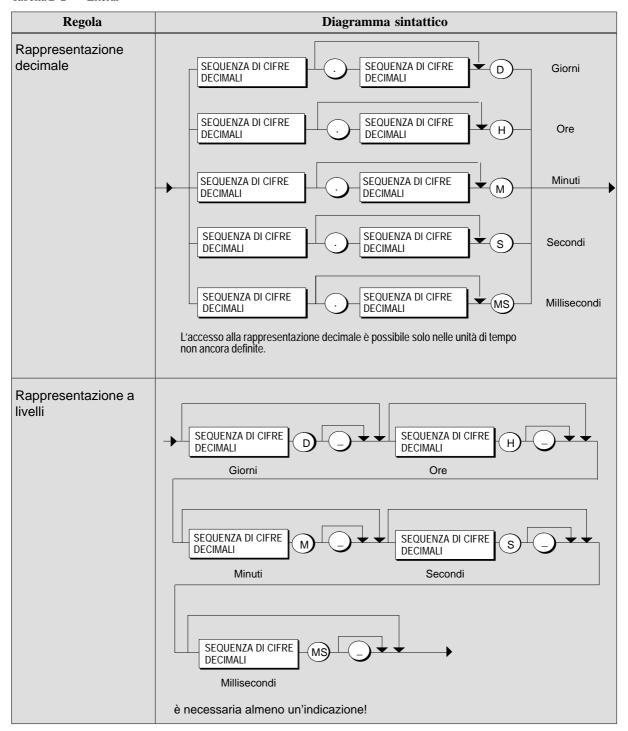
Tabella B-2 Literal

B.1.2 Indirizzamento assoluto

Tabella B-3 Indirizzamento assoluto

Regola	Diagramma sintattico
ACCESSO SEMPLICE ALLA MEMORIA	IDENTIFICAZIONE OPERANDO — Indirizzo — Accesso assoluto; IDENTIFICATORE / SIMBOLO — Accesso simbolico
ACCESSO INDICIZZATO ALLA MEMORIA	IDENTIFICAZIONE OPERANDO — [— Espressione base — , — Espressione base —] ; Indice ; solo in caso di accesso a bit
IDENTIFICAZIONE DELL'OPERANDO PER MEMORIA	Prefisso di memoria — Prefisso di dimensione
ACCESSO ASSOLUTO AL DB	Identificazione operando DB — Indirizzo ; Accesso assoluto
ACCESSO INDICIZZATO AL DB	Identificazione operando DB — [— Espressione base — , — Espressione base —] ; Indice ; solo in caso di accesso a bit
ACCESSO STRUTTURATO AL DB	Identificazione DB — . — Variabile semplice

Tabella B-3 Indirizzamento assoluto

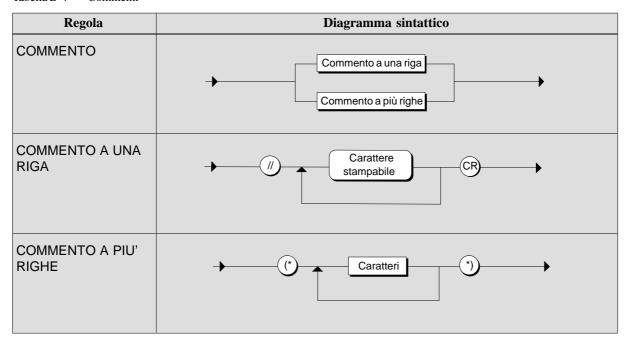
B.2 Commenti

Che cosa si deve osservare

Seguono i punti più importanti da osservare quando si inseriscono dei commenti:

- Non è consentito l’annidamento di commenti.
- L’inserimento di commenti è possibile in qualsiasi posizione nelle regole sintattiche, ma non nelle regole lessicali.

Tabella B-4 Commenti

Regola	Diagramma sintattico
COMMENTO	Commento a una riga / Commento a più righe
COMMENTO A UNA RIGA	// Carattere stampabile CR
COMMENTO A PIU’ RIGHE	(* Caratteri *)

B.3 Attributi di blocco

Che cosa si deve osservare

Gli attributi di blocco possono essere situati con la seguente sintassi dopo l'IDENTIFICAZIONE DEL BLOCCO e prima della dichiarazione del primo blocco di variabili o parametri.

Tabella B-5 Attributi

Regola	Diagramma sintattico
TITOLO	TITLE = ' Carattere stampabile '
VERSIONE	VERSIONE : ' SEQUENZA DI CIFRE DECIMALI (0...15) . SEQUENZA DI CIFRE DECIMALI (0...15) '
PROTEZIONE DEL BLOCCO	KNOW_HOW_PROTECT
AUTORE	AUTHOR : IDENTIFICATORE (max. 8 caratteri)
NOME	NAME : IDENTIFICATORE (max. 8 caratteri)
FAMIGLIA DEL BLOCCO	FAMILY : IDENTIFICATORE (max. 8 caratteri)
Attributi di sistema per blocchi	{ IDENTIFICATORE (max. 24 caratteri) := ' carattere stampabile ' } ;

Regole sintattiche C

Definizione delle regole sintattiche

Sulla base delle regole lessicali, le regole sintattiche descrivono la struttura di SCL. Nell'ambito di tali regole, l'utente può creare il proprio programma SCL in formato libero.

Sommario dell'appendice

Regole formali

Ogni regola ha un nome preposto ad essa. Se la regola viene impiegata in una regola sovraordinata, questo nome viene iscritto in un rettangolo.

Se il nome nel rettangolo è scritto in caratteri maiuscoli si tratta di un token che viene descritto nelle regole lessicali.

Se le regole sono racchiuse in una cornice arrotondata o in un cerchio, le informazioni sul loro contenuto sono riportate in appendice A.

Che cosa si deve osservare

La proprietà formato libero significa per l'utente:

- Si possono inserire caratteri di formattazione in qualsiasi posizione
- Si possono inserire commenti al blocco e commenti alla riga (vedere capitolo 7.6).

C.1 Suddivisione delle sorgenti SCL

Tabella C-1 Sintassi delle sorgenti SCL

Regola	Diagramma sintattico
Programma SCL	Unità di programma SCL
Unità di programma SCL	Blocco organizzativo Funzione Blocco funzionale Blocco dati Tipo di dati definito dall'utente
Blocco organizzativo	ORGANIZATION_BLOCK IDENTIFICAZIONE OB Parte dichiarazioni OB BEGIN Parte istruzioni END_ORGANIZATION_BLOCK
Funzione Si osservi che nelle funzioni senza VOID nella parte istruzioni, il valore di ritorno deve essere assegnato al nome della funzione!	FUNCTION IDENTIFICAZIONE FC : VOID Specificazione tipo di dati Parte dichiarazioni FC BEGIN Parte istruzioni END_FUNCTION
Blocco funzionale	FUNCTION_BLOCK IDENTIFICAZIONE FB Parte dichiarazioni FB BEGIN Parte istruzioni END_FUNCTION_BLOCK

Tabella C-1 Sintassi delle sorgenti SCL

Regola	Diagramma sintattico
Blocco dati	DATA_BLOCK → IDENTIFICAZIONE DB → Parte dichiarazioni DB → BEGIN → Parte assegnazioni DB → END_DATA_BLOCK
Tipo di dato definiti dall'utente	TYPE → IDENTIFICAZIONE UDT → Specificazione tipo di dati STRUCT → END_TYPE

C.2 Struttura della parte dichiarazioni

Tabella C-2 Sintassi della parte dichiarazioni

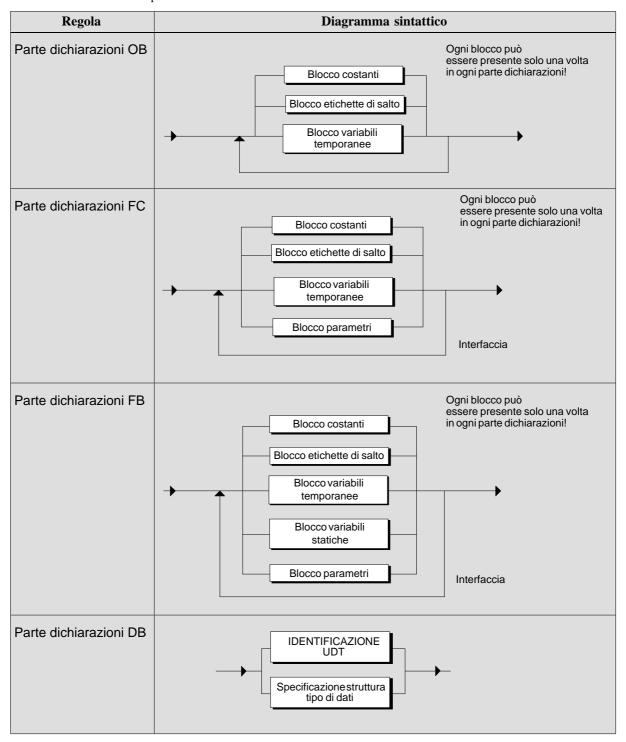



Tabella C-3 Sintassi dei blocchi dichiarazioni

Tabella C-3 Sintassi dei blocchi dichiarazioni

Regola	Diagramma sintattico
Inizializzazione tipo di dati	
Lista d'inizializzazione di campo	
Dichiarazione di istanze	
Blocco variabili temporanee	

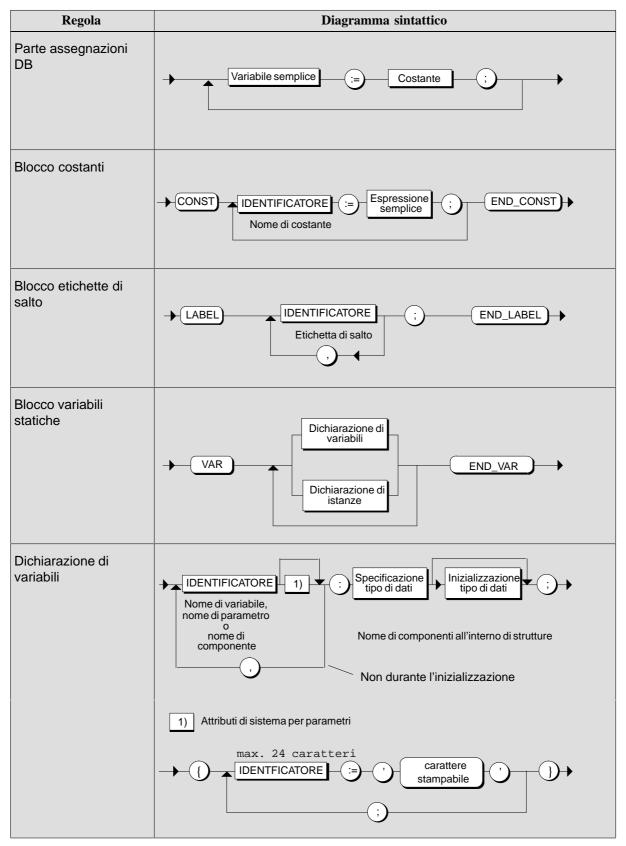
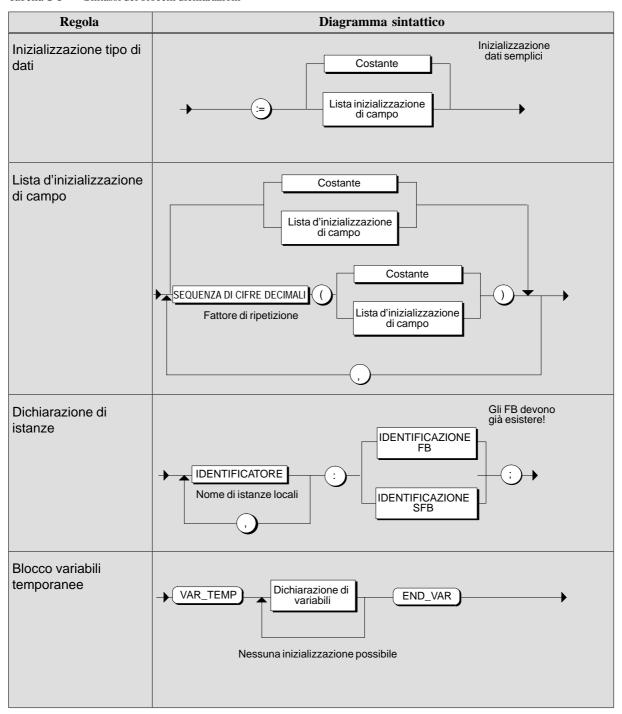



Tabella C-3 Sintassi dei blocchi dichiarazioni

Regola	Diagramma sintattico
Blocco parametri	VAR_INPUT / VAR_OUTPUT / VAR_IN_OUT → Dichiarazione di variabili → END_VAR Inizializzazione possibile solo per VAR_INPUT e VAR_OUTPUT
Specificazione tipo di dati	Tipo di dati semplice DATE_AND_TIME Specificazione Tipo di dati STRINGA Specificazione Tipo di dati ARRAY Specificazione Tipo di dati STRUCT IDENTIFICAZIONE UDT Specificazione Tipo di dati parametri

C.3 Tipi di dati in SCL

Tabella C-4 Sintassi dei tipi di dati nella parte dichiarazioni

Tabella C-4 Sintassi dei tipi di dati nella parte dichiarazioni

Regola	Diagramma sintattico
Tipo di tempo	S5TIME – Tempo, formato S5; TIME – Tempo; TIME_OF_DAY, TOD – Ora del giorno; DATE – Data. vedere anche capitolo B.1.1
DATE_AND_TIME	DATE_AND_TIME# / DT# – Indicazione della data – – – Indicazione dell'ora
Specificazione tipo di dati ARRAY	ARRAY [Index$_1$ .. Index$_n$ (,)] OF Specificazione tipo di dati. Specificazione indice. max. 5 ripetizioni = 6 dimensioni!
Specificazione tipo di dati STRUCT Non dimenticare di mettere un punto e virgola dopo la parola chiave END_STRUCT !	STRUCT Dichiarazione componenti END_STRUCT

Tabella C-4 Sintassi dei tipi di dati nella parte dichiarazioni

Regola	Diagramma sintattico
Dichiarazione componenti	IDENTIFICATORE (Nomi di componenti) → : → Specificazione tipo di dati → [Inizializzazione tipo di dati] → ;
Specificazione tipo di dati parametri	TIMER (Temporizzatore) / COUNTER (Contatore) / ANY (Tipo qualsiasi) / POINTER (Indirizzo) / BLOCK_FC (Funzione) / BLOCK_FB (Blocco funzionale) / BLOCK_DB (Blocco dati) / BLOCK_SDB (Blocco dati di sistema)

C.4 Parte istruzioni

Tabella C-5 Sintassi della parte istruzioni

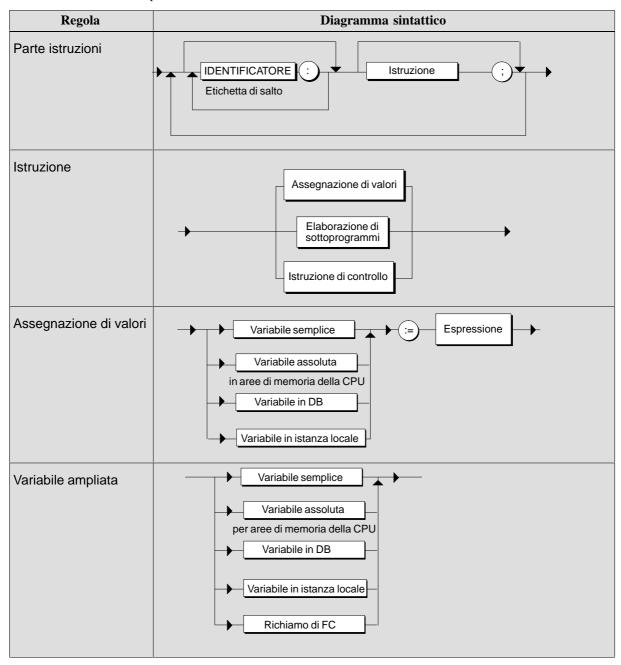



Tabella C-5 Sintassi della parte istruzioni

C.5 Assegnazioni di valori

Tabella C-6 Sintassi delle assegnazioni di valori

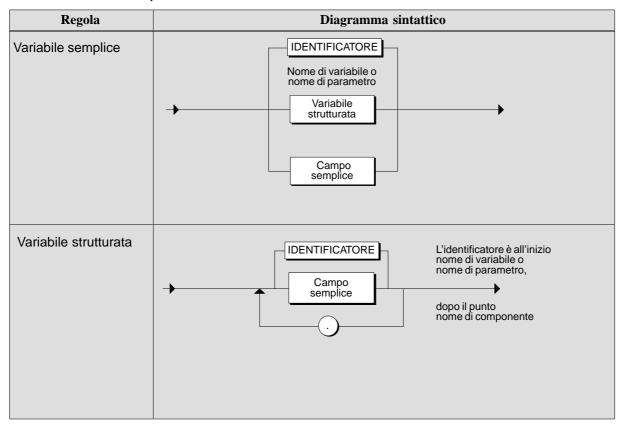



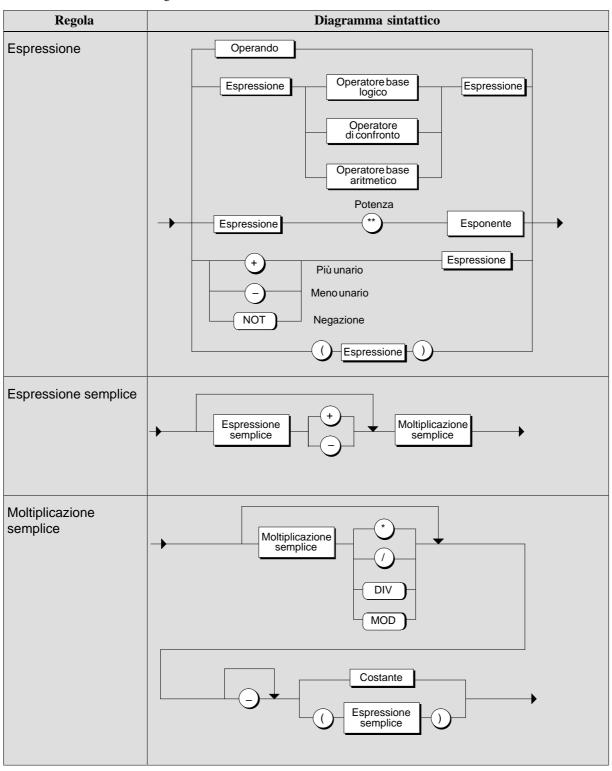
Tabella C-6 Sintassi delle assegnazioni di valori

Regola	Diagramma sintattico
Operando	Costante Variabile ampliata (Espressione) NOT Operando
Variabile ampliata	Variabile semplice Variabile assoluta per aree di memoria della CPU Variabile in DB Variabile in istanza locale Richiamo di FC
Costante	Costante Valore numerico Sequenza di caratteri Nome di costante
Esponente	Variabile ampliata SEQUENZA DI CIFRE DECIMALI (– SEQUENZA DI CIFRE DECIMALI)
Operatore logico di base	AND & XOR OR

Tabella C-6 Sintassi delle assegnazioni di valori

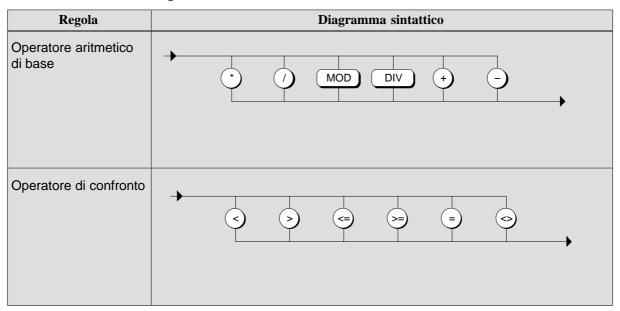
Regola	Diagramma sintattico
Operatore aritmetico di base	* / MOD DIV + –
Operatore di confronto	< > <= >= = <>

C.6 Richiamo di funzioni e blocchi funzionali

Tabella C-7 Sintassi dei richiami

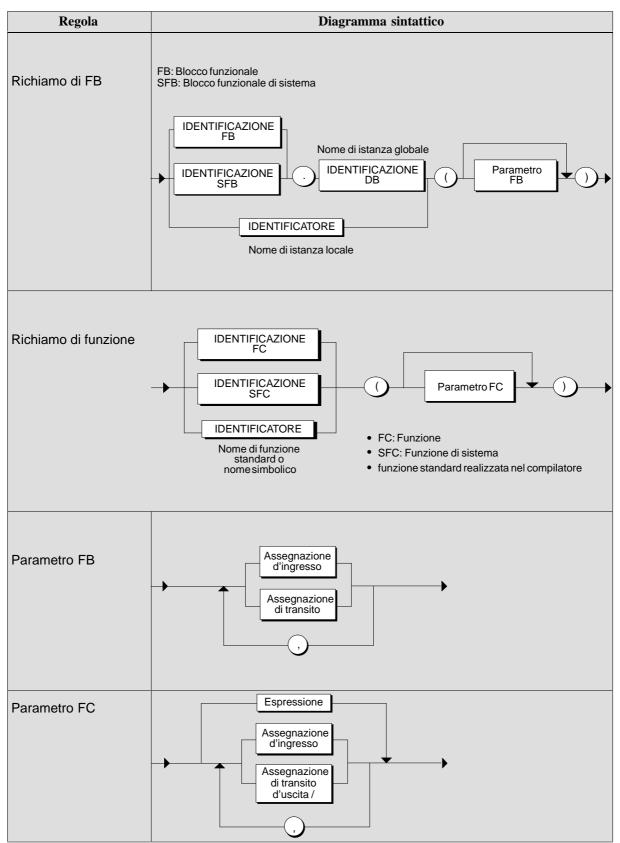



Tabella C-7 Sintassi dei richiami

C.7 Istruzioni di controllo

Tabella C-8 Sintassi delle istruzioni di controllo

Tabella C-8 Sintassi delle istruzioni di controllo

Regola	Diagramma sintattico
Valore	LITERAL INTEGER IDENTIFICATORE *Nome di costante*
Istruzioni di ripetizione e istruzioni di salto	Istruzione FOR Istruzione WHILE Istruzione REPEAT Istruzione CONTINUE Istruzione EXIT Istruzione RETURN Istruzione GOTO
Istruzione FOR Non dimenticare di mettere un punto e virgola dopo la parola chiave END_FOR.	FOR Assegnazione iniziale TO Espressione base per valore finale BY Espressione base per ampiezza di passo DO Parte assegnazioni END_FOR
Assegnazione iniziale	Variabile semplice del tipo di dati INT/DINT := Espressione base per valore iniziale

Tabella C-8 Sintassi delle istruzioni di controllo

Regola	Diagramma sintattico
Istruzione WHILE Non dimenticare di mettere un punto e virgola dopo la parola chiave END_WHILE.	WHILE Espressione DO Parte istruzioni END_WHILE
Istruzione REPEAT Non dimenticare di mettere un punto e virgola dopo la parola chiave END_REPEAT.	REPEAT Parte istruzioni UNTIL Espressione END_REPEAT
Istruzione CONTINUE	CONTINUE
Istruzione RETURN	RETURN
Istruzione EXIT	EXIT
Salto di programma	GOTO IDENTIFICATORE Etichetta di salto

Bibliografia D

/12/ Descrizione: *Sistema di automazione S7-300*
Struttura ed impiego

/13/ Descrizione: *Sistema di automazione S7-400*
Struttura ed impiego

/14/ Descrizione: *Microcomputer industriali M7-300/400*
Struttura ed impiego

/20/ Descrizione: *Sistema di automazione S7-300/400*
Programmazione

/25/ Descrizione: *Microcomputer industriali M7*
Programmazione

/30/ Prontuario: *Sistema di automazione S7-300*
Introduzione alla configurazione e programmazione

/70/ Manuale: *Sistema di automazione S7-300*
Installazione, configurazione e dati della CPU

/71/ Manuale di riferimento: *Sistemi di automazione S7-300, M7-300*
Caratteristiche delle unità modulari

/72/ Lista operazioni: *Sistema di automazione S7-300*

/100/ Manuale: *Sistemi di automazione S7-400, M7-400*
Configurazione

/101/ Manuale di riferimento: *Sistemi di automazione S7-400, M7-400*
Caratteristiche delle unità modulari

/102/ Lista operazioni: *Sistema di automazione S7-400*

/230/ Manuale di conversione: STEP 7
Da S5 a S7

/231/ Manuale utente: *Software di base per S7 e M7*
STEP 7

/232/ Manuale: *AWL per S7-300/400*
Programmazione di blocchi

/233/ Manuale: *KOP per S7-300/400*
Programmazione di blocchi

/234/ Manuale di programmazione: *Software di sistema per S7-300/400*
Sviluppo di programmi

/235/ Manuale di riferimento: *Software di sistema per S7-300/400*
Funzioni standard e di sistema

/236/ Manuale: *FUP per S7-300/400*
Programmazione di blocchi

/237/ *Indice generale*, STEP 7

/249/ Manuale: *CFC Continuous Function Chart*
Volume 2

/251/ Manuale: *GRAPH per S7-300/400*
Programmazione di comandi sequenziali

/252/ Manuale: *HiGRAPH per S7-300/400*
Programmazione di diagrammi di stato

/253/ Manuale: *C per S7-300/400*
Programmazione in C

/254/ Manuale: *CFC Continuos Chart*
Volume 1

/290/ Manuale utente: *ProC/C++ per M7-300/400*
Programmazione in C

/291/ Manuale utente: *ProC/C++ per M7-300/400*
Debugger per programmi in C

/800/ DOCPRO,
Sviluppo di schemi circuitali a norma (solo su CD)

/800/ Manuale di riferimento: *Software di sistema per S7-300/400*
STEP 7 Funzioni standard, parte 2 (solo su CD)

Glossario

A

ALT

Lo stato di funzionamento ALT viene raggiunto dallo stato di funzionamento RUN mediante richiesta del dispositivo di programmazione. In questo stato di funzionamento sono possibili speciali funzioni di test.

Area di memoria

In SIMATIC S7, un'unità centrale possiede tre aree di memoria: la memoria di caricamento, la memoria di lavoro e l'area di sistema.

Attributo

Un attributo è una proprietà che può essere agganciata p. es. ad un'identificazione di blocco o ad un nome di variabile. In SCL esistono attributi per le seguenti indicazioni: titolo blocco, versione, protezione blocco, autore, nome di blocco, famiglia di blocco.

B

Blocco

Per la loro funzione, struttura e scopi d'impiego, i blocchi sono parti ben definite di un programma utente. In STEP 7 esistono blocchi di codice (FB, FC, OB, SFC, SFB), blocchi dati (DB, SDB) e tipi di dati definiti dall'utente (UDT).

Blocco dati di istanza (DB di istanza)

Un blocco dati di istanza memorizza i parametri formali e i dati statici locali dei blocchi funzionali. Un blocco dati di istanza può essere assegnato ad un richiamo di FB o ad una gerarchia di richiamo di blocchi funzionali. Esso viene generato automaticamente in SCL.

Blocco dati di sistema (SDB)

I blocchi dati di sistema sono aree di dati nell'unità centrale contenenti impostazioni di sistema e parametri di unità. I blocchi dati di sistema vengono generati e modificati con il software di base STEP 7.

Blocco di codice

In SIMATIC S7, un blocco di codice è un blocco che contiene una parte del programma utente STEP 7. Un blocco dati contiene invece esclusivamente dei dati. Esistono i seguenti blocchi di codice: blocchi organizzativi (OB), blocchi funzionali (FB), funzioni (FC), blocchi funzionali di sistema (SFB) e funzioni di sistema (SFC).

Blocco dati (DB)

I blocchi dati (DB) sono aree di dati del programma utente che contengono dati utente. Ai blocchi dati si può accedere da tutti i blocchi di codice. I blocchi dati assegnati ad un determinato richiamo di FB vengono denominati blocchi dati di istanza.

Blocco funzionale (FB)

Conformemente a IEC 1131-3, un blocco funzionale (FB) è un blocco di codice con dati statici. Un FB consente di trasferire dei parametri nel programma utente. Perciò, i blocchi funzionali sono particolarmente adatti per programmare funzioni complesse e di frequente ripetizione, come p. es. regolazioni, scelta del modo operativo. Poiché un FB dispone di una memoria (blocco dati di istanza), si può accedere ai suoi parametri (p. es. uscite) in qualsiasi momento e in qualsiasi punto del programma utente.

Blocco funzionale di sistema (SFB)

Un blocco funzionale di sistema (SFB) è un blocco funzionale integrato nel sistema operativo della CPU, che in caso di necessità può essere richiamato nel programma utente STEP 7.

Blocco organizzativo (OB)

I blocchi organizzativi costituiscono l'interfaccia fra il sistema operativo della CPU e il programma utente. Nei blocchi organizzativi viene definita la sequenza di elaborazione del programma utente.

C

Campo

Un campo (ARRAY) è un tipo di dati composto, contenente elementi di dati dello stesso tipo. Questi elementi di dati possono a loro volta essere del tipo semplice o composto.

Caricamento nel PG

Caricamento di oggetti caricabili (p. es. blocchi di codice) dalla memoria di caricamento di un'unità programmabile nel dispositivo di programmazione. Ciò può aver luogo sia tramite un dispositivo di programmazione direttamente collegato sia p. es. mediante il PROFIBUS.

Caricamento nel sistema di destinazione

Caricamento di oggetti caricabili (p. es. blocchi di codice) dal dispositivo di programmazione nella memoria di caricamento di un'unità programmabile. Ciò può aver luogo sia tramite un dispositivo di programmazione direttamente collegato sia p. es. mediante il PROFIBUS.

Classe di blocchi

I blocchi vengono suddivisi in due classi in base al loro contenuto: blocchi di codice e blocchi di dati.

Commento a più righe

Informazioni supplementari relative a più righe (p. es. spiegazioni concernenti il processo automatizzato), che non possono essere caricate nella memoria di lavoro dei sistemi di automazione SIMATIC S7.

Compilazione

Generazione di un programma utente eseguibile partendo da una sorgente.

Compilazione orientata alla sorgente

Nell'introduzione orientata alla sorgente, durante la compilazione viene eseguito un controllo per accertare eventuali errori di introduzione. Un codice eseguibile viene generato solo se non vi sono più errori.

Compilatore SCL

Il compilatore SCL è un compiler batch, con il quale il programma editato in precedenza (sorgente SCL) viene compilato in codice macchina MC7. I blocchi così generati vengono depositati nel programma S7 nel contenitore "Blocchi".

Contatori

I contatori sono componenti della memoria di sistema della CPU. Il contenuto di questi contatori viene attualizzato in modo asincrono con il programma utente del sistema operativo. Con istruzioni STEP 7 viene definita la funzione dettagliata della cella di conteggio (p. es. contatore in avanti) e viene avviata la sua elaborazione (Start).

Contenitore

Raccoglitore dell'interfaccia operativa del SIMATIC Manager, che può essere aperto e può contenere ulteriori contenitori e oggetti.

Controllo continuativo

Modo di test di SCL. Durante il controllo continuativo di un programma si può testare un gruppo di istruzioni. Il gruppo di istruzioni viene denominato anche area di controllo.

Conversione esplicita

La conversione esplicita signfica inserire una funzione di conversione nel programma sorgente. Quando si combinano due operandi di tipo diverso, l'utente deve eseguire una corrispondente conversione. Quando si passa ad un'altra classe, p. es. dal tipo dati a bit al tipo dati numerici e – quando il tipo dati di destinazione è meno importante del tipo dati di sorgente – anche durante il cambio all'interno di una classe dello stesso tipo.

Conversione implicita

La conversione implicita significa che una funzione di conversione viene inserita automaticamente dal compilatore. Quando si combinano due operandi di tipo diverso viene eseguita una conversione: se non vi è alcun cambio ad un'altra classe e se il tipo di dati di destinazione non è meno importante del tipo dati di sorgente.

Costante (literal)

Sono costanti il cui valore e tipo viene determinato dal modo di scrittura formale. Si distingue tra literal numerici, literal di caratteri e literal per indicazioni di tempo.

Costante (simbolica)

Le costanti con nomi simbolici sono segnaposti per valori costanti di blocchi di codice. Le costanti simboliche vengono utilizzate per facilitare la lettura di un programma.

D

Dati globali

I dati globali sono dati ai quali si può far riferimento da qualsiasi blocco di codice (FC, FB, OB). Si tratta in particolare di merker M, ingressi E, uscite A, temporizzatori, contatori ed elementi di blocchi dati DB. Ai dati globali si può accedere sia in modo assoluto che simbolico.

Dati locali

I dati locali sono i dati assegnati ad un blocco di codice e che vengono dichiarati nella sua parte di dichiarazione o nella parte dichiarazioni. Essi comprendono (a seconda del tipo di blocco): parametri formali, dati statici, dati temporanei.

Dati statici

I dati statici sono dati locali di un blocco funzionale che vengono memorizzati nel blocco dati di istanza e perciò rimangono immutati fino alla successiva elaborazione del blocco funzionale.

Dati temporanei

I dati temporanei a livello locale appartengono al blocco di codice e **non occupano** alcuna area di memoria statica poiché essi vengono depositati nello stack della CPU. I loro valori rimangono immutati solo durante l'esecuzione di un blocco.

Dati utili

I dati utili vengono scambiati fra un'unità centrale e un'unità di segnale, fra un'unità funzionale e unità di comunicazione tramite l'immagine di processo o accessi diretti. I dati utili possono essere: segnali di ingresso/uscita digitali e analogici, informazioni di comando e di stato di unità funzionali.

Debugger SCL

Il Debugger SCL è un debugger per linguaggi avanzati con il quale si possono localizzare errori logici di programmazione nei programmi utente creati in SCL.

Decompilazione

Tramite la decompilazione conformemente alla rappresentazione AWL, è possibile caricare e visualizzare il blocco caricato nella CPU con qualsiasi PG/PC. Possono anche essere assenti determinate parti del blocco come p.es. simboli e commenti.

Dichiarazione dei tipi dei dati

Con la dichiarazione dei tipi dei dati l'utente può dichiarare tipi di dati definiti dall'utente.

Dichiarazione di variabili

La dichiarazione di variabili comprende l'indicazione di un nome simbolico, di un tipo di dati, eventualmente di un valore di preassegnazione, indirizzo e commento.

E

Editazione orientata alla sorgente

La programmazione in SCL consente l'introduzione orientata alla sorgente di un programma STEP 7. L'introduzione di un programma può essere eseguita con qualsiasi tipo di editor. Il codice programma vero e proprio viene generato solo durante la compilazione. In questa fase vengono identificati anche eventuali errorri. Questo tipo di introduzione è particolarmente adatto per la creazione simbolica di programmi standard.

Editor SCL

L'Editor SCL è un editor adattato a SCL con il quale si può creare una sorgente SCL.

Enable (EN)

In STEP 7, ogni blocco possiede un ingresso ”Enable” (EN), che può essere impostato durante il richiamo di un blocco. Se su EN è presente un segnale 1, il blocco viene richiamato, mentre se è presente un segnale 0 il blocco non viene richiamato.

Enable out (ENO)

In STEP 7, ogni blocco possiede un'uscita ”Enable Output” (ENO). All'interno del blocco, l'utente può associare l'ingresso ”Enable” con un valore interno (AND). Il risultato viene assegnato automaticamente all'uscita ENO. Nei richiami concatenati di blocchi, con ENO si può eseguire l'elaborazione dei blocchi successivi in funzione della corretta elaborazione del blocco precedente.

Espressione

In SCL un'espressione serve per l'elaborazione dei dati. Si distingue fra espressioni aritmetiche, logiche e di confronto.

F

Funzione (FC)

Conformemente a IEC 1131-3, una funzione (FC) è un blocco di codice **senza** dati statici. Una funzione offre la possibilità di trasferire parametri nel programma utente. In tal modo, le funzioni sono particolarmente adatte per parametrizzare funzioni complesse di frequente ripetizione, come p. es. calcoli.

Funzione di sistema (SFC)

Una funzione di sistema (SFC) è una funzione integrata nel sistema operativo della CPU, che in caso di necessità può essere richiamata nel programma utente STEP 7.

G

Gerarchia di richiamo

Prima che possano essere elaborati, tutti i blocchi devono dapprima essere richiamati. La sequenza e l'annidamento di questi richiami viene denominata gerarchia di richiamo.

Guida online

STEP 7 offre all'utente la possibilità di far visualizzare sullo schermo testi ausiliari dipendenti dal contesto mentre si lavora con il software di programmazione.

I

Identificatore

Il termine identificatore viene usato per far riferimento agli oggetti linguistici di SCL. Esistono le seguenti classi: identificatori standard, nomi predefiniti e parole chiave, identificatori assoluti, (o identificazioni di operando), nomi selezionabili liberamente, p. es. per variabili ed etichette di salto, oppure nomi simbolici generati in una tabella dei simboli.

Identificazione di operando

Un'identificazione di operando è una parte dell'operando di un'operazione contenente informazioni, come p. es. l'area di memoria in cui l'operazione trova un valore (oggetto dati) con cui esegue una combinazione oppure trova la grandezza di un valore (oggetto dati) con il quale esegue una combinazione. Nell'istruzione "Valore := EB10" "EB" è l'identificazione di operando ("E" indica l'area ingresso della memoria, "B" indica un byte in quest'area).

Immagine di processo

Nella CPU, gli stati dei segnali delle unità di ingresso e uscita digitali vengono trasferiti in un'immagine di processo. Si distingue fra l'immagine di processo degli ingressi (IPI) e quelle delle uscite (IPU).

Immagine di processo delle uscite (IPU)

Alla fine del programma utente, l'immagine di processo delle uscite viene trasferita dal sistema operativo alle unità di uscita.

Immagine di processo degli ingressi (IPI)

Prima dell'elaborazione del programma utente, il sistema operativo legge l'immagine di processo degli ingressi dalle unità d'ingresso.

Indirizzamento assoluto

Nell'indirizzamento assoluto viene indicato l'indirizzo dell'operando da elaborare. Esempio: l'indirizzo A 4.0 indica il bit 0 nel byte 4 dell'immagine di processo delle uscite.

Indirizzamento simbolico

Nell'indirizzamento simbolico, l'operando da elaborare viene indicato in modo simbolico (al posto di un indirizzo).

Integer (INT)

Integer (INT) è uno dei tipi di dati semplici. La rappresentazione avviene con un numero intero a 16 bit.

Interfaccia di richiamo

L'interfaccia di richiamo viene definita tramite i parametri d'ingresso, d'uscita e di transito (parametri formali) di un blocco nel programma utente STEP 7. Al richiamo del blocco, questi parametri vengono sostituiti dai parametri attuali.

Istanza

Con il termine "Istanza" si indica il richiamo di un blocco funzionale. Ad esso viene assegnato un blocco dati di istanza oppure un'istanza locale. Se un blocco funzionale nel programma utente STEP 7 viene richiamato n volte con diversi parametri e nomi di blocchi dati di istanza, esistono n istanze

```
FB13.DB3 (P3:=...), FB13.DB4 (P4:=...),
FB13.DB5 (P5:=...), .....FB13.DBn (Pn:=...).
```

Istanza locale

Un'istanza locale viene definita nella parte delle variabili statiche di un blocco funzionale. Al posto di un intero blocco dati di istanza viene utilizzata solo una parte locale come area dati per il blocco funzionale e cioè quella parte che viene richiamata con il nome di istanza locale.

Istruzione

Un'istruzione è la più piccola unità indipendente di un programma utente creato in un linguaggio testuale. Essa rappresenta una norma di lavoro per il processore.

Istruzione CASE

Questa istruzione è un'istruzione di diramazione. Essa serve, in funzione del valore di un'espressione di selezione, per effettuare la scelta di una parte del programma da 1 a n.

Istruzione CONTINUE

Conclude una variabile d'esecuzione e la avvia con il valore successivo della variabile d'esecuzione.

Istruzione EXIT

Interruzione di un loop d'esecuzione.

Istruzione FOR

Un'istruzione FOR serve per ripetere una sequenza di istruzioni finché la variabile d'esecuzione si mantiene nell'ambito dei valori prescritti.

Istruzione GOTO

Un'istruzione GOTO provoca il salto immediato ad una determinata etichetta.

Istruzione REPEAT

Un'istruzione REPEAT serve per ripetere una sequenza di istruzioni finché non si verifica una condizione d'interruzione.

Istruzione RETURN

Questa istruzione provoca l'uscita dal blocco attuale.

L

Lista istruzioni (AWL)

La lista istruzioni (AWL) è un linguaggio di programmazione testuale a livello macchina.

M

Memoria di sistema (area di sistema)

La memoria di sistema è integrata nell'unità centrale ed ha una struttura di memoria RAM. Nella memoria di sistema vengono depositate aree di operandi (p. es. temporizzatori, contatori, merker) e aree di dati usati dal sistema operativo per operazioni interne (p. es. buffer di comunicazione).

Merker (M)

Area di memoria nella memoria di sistema di una CPU S7 SIMATIC. A quest'area si può accedere in lettura e scrittura (a bit, a byte, a parola e a doppia parola). L'area merker può essere utilizzata dall'utente per memorizzare risultati intermedi.

Mnemonico

Il mnemonico è una rappresentazione abbreviata degli operandi e delle operazioni di programmazione in un programma (p. es. "E" significa ingresso). STEP 7 supporta la rappresentazione IEC (che si basa sulla lingua inglese) e la rappresentazione SIMATIC (che si basa sulla rappresentazione tedesca delle operazioni e sulle convenzioni per l'indirizzamento SIMATIC).

Multiistanza

Quando si utilizza la multiistanza, il blocco dati di istanza contiene i dati di molti blocchi funzionali di una gerarchia di richiamo.

N

Non-terminale

Un non-terminale è un elemento composto che viene descritto da un'ulteriore regola lessicale o sintattica.

Numero in virgola mobile

Un numero in virgola mobile è un numero positivo o negativo contenente un valore decimale come p. es. 0.339 o -11.1.

O

Offline

Offline indica lo stato di funzionamento in cui il dispositivo di programmazione non ha alcun collegamento con il sistema di automazione (fisico, logico).

Online

Online indica lo stato di funzionamento in cui il dispositivo di programmazione è collegato con il sistema di automazione (fisico, logico).

Operando

Un operando è una parte di un'istruzione e indica con che cosa deve operare il processore. Esso può essere indirizzato sia in modo assoluto che in modo simbolico.

Operazione

Un'operazione fa parte di un'istruzione e indica che cosa deve fare il processore.

P

Parametri d'ingresso (parametri E)

I parametri d'ingresso esistono solo nelle funzioni e nei blocchi funzionali. Con l'ausilio dei parametri d'ingresso, i dati vengono trasferiti al blocco richiamante per un'ulteriore elaborazione.

Parametri di transito (parametri D)

I parametri di transito sono presenti nelle funzioni e nei blocchi funzionali. Tramite parametri di transito i dati vengono trasferiti al blocco richiamato, elaborati e il blocco richiamato deposita nuovamente i risultati nella stessa variabile.

Parametri d'uscita (parametri A)

Tramite i parametri d'uscita di un blocco nel programma utente STEP 7 i risultati vengono trasferiti al blocco richiamante.

Parametri formali

Un parametro formale è un segnaposto per il parametro ”effettivo” (parametro attuale) nei blocchi di codice parametrizzabili. Negli FB e nelle FC, i parametri formali vengono dichiarati dall’utente, negli SFB e nelle SFC essi sono già presenti. Durante il richiamo del blocco, al parametro formale viene assegnato un parametro attuale, dopodiché il blocco richiamato lavora con questo valore aggiornato. I parametri formali fanno parte dei dati locali del blocco e vengono suddivisi in parametri d’ingresso, d’uscita e di transito.

Parametro

In SCL: variabile di un blocco di codice (parametro attuale, parametro formale)

Parametro attuale

In un richiamo di un blocco funzionale (FB) o di una funzione (FC), i parametri attuali sostituiscono i parametri formali.

Esempio: il parametro formale ”Start” viene sostituito con il parametro attuale ”E 3.6”

Parola chiave

In SCL, le parole chiave vengono utilizzate per contrassegnare l’inizio di un blocco, per marcare sezioni nella parte di dichiarazione risp. nella parte convenzioni e per contrassegnare delle istruzioni. Esse vengono inoltre utilizzate per commenti e attributi.

Parola di stato

La parola di stato fa parte integrante dei registri dell’unità centrale. La parola di stato contiene informazioni di stato e informazioni di errore che possono verificarsi nel corso dell’elaborazione di comandi STEP 7. I bit di stato possono essere letti e scritti dall’utente; i bit di errore possono solo essere letti.

Parte dichiarazioni

Qui vengono dichiarati i dati locali di un blocco di codice.

Passo singolo

Il passo singolo è un passo di test nell’ambito della funzione passo singolo del Debugger SCL. Nella funzione passo singolo, si può eseguire il programma istruzione per istruzione e visualizzare i risultati nella finestra dei risultati.

Progetto

Un contenitore per tutti gli oggetti di una soluzione di automazione indipendentemente dal numero di stazioni, unità e dal loro impiego.

Programma utente

Il programma utente contiene tutte le istruzioni e le dichiarazioni per l’elaborazione dei segnali, tramite i quali viene controllato l’impianto o un processo. Esso viene assegnato ad un’unità modulare programmabile (p. es. CPU, FM) e può essere strutturato in unità più piccole (blocchi).

Programma utente S7

Il programma utente S7 si trova nel contenitore ”Blocchi”. Esso contiene blocchi, i quali vengono caricati e possono essere eseguiti in un’unità programmabile S7 (p. es. CPU) per comandare un impianto o un processo.

Programmazione simbolica

Il linguaggio di programmazione SCL consente l'impiego di sequenze simboliche di caratteri al posto di operandi: p. es. l'operando A1.1 può essere sostituito da "Valvola_17". La tabella dei simboli in STEP 7 assicura il collegamento fra l'operando e la sequenza simbolica di caratteri assegnati.

Programmazione strutturata

Per risolvere problemi di automazione piuttosto complessi, il programma utente viene suddiviso in piccole parti di programma completamente autonome (blocchi). La suddivisione del programma utente avviene in modo funzionale oppure conformemente alla struttura tecnologica dell'impianto.

Protezione di blocco

Come protezione di blocco viene definita la possibilità di proteggere blocchi singoli contro la decompilazione, se la compilazione della sorgente del blocco è stata eseguita con la parola chiave "KNOW_HOW_PROTECTED".

Punto d'arresto

Con questa funzione, in determinati punti del programma si può portare l'unità centrale (CPU) nello stato di funzionamento ALT. Quando viene raggiunto un punto d'arresto, si possono eseguire le funzioni di test come p. es. elaborazione passo passo dei comandi oppure comando/controllo di variabili.

R

Rappresentazione BCD

In STEP 7, all'interno della CPU l'indicazione di temporizzatori e contatori avviene solo in formato BCD. BCD significa "Codice binario per cifre decimali".

Regola sintattica

Il livello di regole superiore relativo alla descrizione formale del linguaggio SCL si compone di regole sintattiche. Per il loro impiego esiste libertà di formato, ovvero p. es. si possono inserire spazi e caratteri di controllo.

Regole lessicali

Il livello di regole inferiore relativo alla descrizione formale del linguaggio SCL si compone di regole lessicali. Per il loro uso non esiste libertà di formato, ovvero l'integrazione con spazi e caratteri di controllo (p. es.) non è consentita.

Richiamo di blocco

Avvio di un blocco nel programma utente STEP 7: in linea di massima, i blocchi organizzativi vengono richiamati dal sistema operativo, tutti gli altri blocchi vengono richiamati dal programma utente STEP 7.

RUN

Nello stato di funzionamento RUN viene elaborato il programma utente e l'immagine di processo viene aggiornata ciclicamente. Tutte le uscite digitali sono abilitate.

RUN-P

Lo stato di funzionamento RUN-P corrisponde allo stato di funzionamento RUN, con la differenza che nelle stato di funzionamento RUN-P sono consentite tutte le funzioni del dispositivo di programmazione senza alcuna restrizione.

S

SCL
Linguaggio di programmazione avanzato, simile a PASCAL secondo la norma DIN EN-61131-3 (int. IEC 1131-3) per la programmazione di compiti complessi in un PLC, p. es. algoritmi, compiti di elaborazione dati. Abbreviazione di ”Structured Control Language”.

Simbolo
Un simbolo è un nome definito dall’utente conformemente a determinate regole sintattiche. Dopo la definizione indicante il suo tipo (p. es. variabile, tipo di dati, blocco), questo nome può essere utilizzato in fase di programmazione e nelle operazioni di comando e osservazione.
Esempio: operando: E 5.0, tipo di dati: Bool, simbolo: Tasto_Arresto di emergenza.

Single Step
⇒Passo singolo

Sorgente
Una sorgente (file di testo) contiene il codice sorgente (testo ASCII) che può essere creato con qualsiasi Editor. Una sorgente viene compilata con un compilatore (AWL, SCL) in un programma utente eseguibile. Una sorgente viene depositata nel contenitore ”Sorgenti” del programma S7.

Sorgente SCL
La sorgente SCL è il file in cui viene creato il programma in SCL. La sorgente viene quindi compilata con il compilatore SCL.

Stato del blocco
⇒ Controllo continuativo.

Struttura (STRUCT)
Una struttura è un tipo di dati composto e contiene elementi di dati di vario tipo. Questi elementi di dati possono essere del tipo semplice o composto.

T

Tabella dei simboli
Tabella per l’assegnazione di simboli (=nome) a indirizzi di dati globali e blocchi. Esempi: Arresto di emergenza (simbolo) – E 1.7 (indirizzo) o Regolatore (simbolo) – SFB 24 (blocco).

Tabella delle variabili
Nella tabella delle variabili vengono riunite le variabili che devono essere osservate e comandate, comprese le corrispondenti indicazioni sui formati.

Tempo di ciclo
Il tempo di ciclo è il tempo di cui ha bisogno la CPU per l’elaborazione del programma utente.

Tempo di controllo del ciclo
Se il tempo di elaborazione del programma utente supera il tempo di controllo del ciclo impostato in precedenza, il sistema operativo genera un messaggio di errore e la CPU si porta in STOP.

Temporizzatori

I temporizzatori sono componenti della memoria di sistema della CPU. Il contenuto di questi temporizzatori viene aggiornato in modo asincrono rispetto al programma utente dal sistema operativo. Con istruzioni STEP 7 viene definita la funzione dettagliata della cella di tempo (p. es. ritardo all'inserzione) e viene avviata la sua elaborazione (Start).

Terminale

Un terminale è un elemento base di una regola lessicale o sintattica, il quale non viene dichiarato con un un'ulteriore regola, ma solo verbalmente. Un terminale può essere p. es. una parola chiave o solo un singolo carattere.

Tipo di dati

Con l'ausilio del tipo di dati si definisce in che modo il valore di una variabile o costante deve essere utilizzato in un programma utente. In SCL, l'utente dispone di tre tipi di dati:

- Tipi di dati semplice
- Tipo di dati composto
- Tipi di dati definiti dall'utente (UDT).

Tipi di dati composti

Si distingue fra strutture e campi. Le "strutture" vengono composte con diversi tipi di dati (p. es. tipi di dati semplici). I "campi" si compongono di diversi elementi di uno stesso tipo di dati. Anche i tipi di dati `STRING` e `DATE_AND_TIME` sono tipi di dati composti.

Tipi di dati definiti dall'utente

I tipi di dati definiti dall'utente (UDT) vengono creati dall'utente con la dichiarazione dei tipi di dati. Essi possiedono un nome proprio e possono essere utilizzati più volte. In tal modo, un tipo di dati definito dall'utente può essere utilizzato per generare diversi blocchi dati con identica struttura (p. es. regolatore).

Tipi di dati semplici

I tipi di dati semplici sono tipi di dati predefiniti conformemente a IEC 1131-3. Esempi: il tipo di dati "`BOOL`" definisce una variabile binaria ("`Bit`"); il tipo di dati "`INT`" definisce una variabile in virgola fissa a 16 bit.

Tipo di blocco

L'architettura dei blocchi di STEP 7 conosce i seguenti tipi di blocchi: blocchi organizzativi, blocchi, funzionali, funzioni, blocchi dati e blocchi funzionali di sistema, funzioni di sistema, blocchi dati di sistema e tipi di dati definiti dall'utente ⇒ Blocco.

Tipo di dichiarazione

Il tipo di dichiarazione indica in che modo un parametro o una variabile locale deve essere utilizzato da un blocco. Esistono parametri d'ingresso, parametri d'uscita e parametri di transito, così come variabili statiche e temporanee.

Tipo di parametro

Un tipo di parametro è uno tipo di dati speciale per temporizzatori, contatori e blocchi. Esso può essere utilizzato nei parametri d'ingresso di blocchi funzionali **e** funzioni, mentre i parametri di transito possono essere usati solo da blocchi funzionali, per trasferire temporizzatori, contatori e blocchi al blocco richiamato.

U

UDT

⇒ Tipi di dati definiti dall’utente.

V

Variabile

Una variabile definisce dei dati con contenuto variabile, i quali possono essere utilizzati nel programma utente STEP 7. Una variabile si compone di un operanado (p. es. M 3.1) e di un tipo di dati (p. es. Bool) e può essere contrassegnata con un simbolo (p. es. NASTRO_ON). La variabile viene dichiarata nella parte dichiarazioni.

Variabile OK

La variabile OK serve per annotare l’esecuzione corretta o errata di una sequenza di comando blocchi. Si tratta di una variabile globale del tipo BOOL.

Indice analitico

R

S

T

U

V

W

Siemens AG
A&D AS E46
Östliche Rheinbrückenstr. 50

D-76181 Karlsruhe
Repubblica federale di Germania

Mittente :

Nome: _
Funzione: _
Ditta: _
Via: _
C.A.P.: _
Città: _
Paese: _
Telefono: _

Indicare il corrispondente settore industriale:

- ❒ Industria automobilistica
- ❒ Industria chimica
- ❒ Industria elettrotecnica
- ❒ Industria alimentare
- ❒ Tecnica di controllo e strumentazione
- ❒ Industria meccanica
- ❒ Petrolchimica
- ❒ Industria farmaceutica
- ❒ Industria delle materie plastiche
- ❒ Industria cartaria
- ❒ Industria tessile
- ❒ Impresa di trasporti
- ❒ Altre _ _ _ _ _ _ _ _ _ _ _

Critiche/suggerimenti

Vi preghiamo di volerci comunicare critiche e suggerimenti atti a migliorare la qualità e, quindi, a facilitare l’uso della documentazione. Per questo motivo Vi saremmo grati se vorreste compilare e spedire alla Siemens il seguente questionario.

Servendosi di una scala di valori da 1 per buono a 5 per scadente, Vi preghiamo di dare una valutazione sulla qualità del manuale rispondendo alle seguenti domande.

1. Corrisponde alle Vostre esigenze il contenuto del manuale? ☐
2. È facile trovare le informazioni necessarie? ☐
3. Le informazioni sono spiegate in modo sufficientemente chiaro? ☐
4. Corrisponde alle Vostre esigenze il livello delle informazioni tecniche? ☐
5. Come valutate la qualità delle illustrazioni e delle tabelle? ☐

Se avete riscontrato dei problemi di ordine pratico, Vi preghiamo di delucidarli nelle seguenti righe:

_ _

_ _

_ _

_ _

_ _

_ _

_ _

_ _

_ _

_ _

_ _

_ _